TEKNILLINEN KORKEAKOULU
Tietotekniikan osasto

Timo Ratilainen

Laadunvalvontakorttien visualisointityökalun toteutus laboratorioiden tiedonhallintajärjestelmään

Diplomityö, joka on jätetty opinnäytteenä tarkastettavaksi diplomi-insinöörin tutkintoa varten Espoossa 14.10.2004

Työn valvoja

Professori Reijo Sulonen

Työn ohjaaja

DI Raimo Oinas

Teknillinen korkeakoulu **Diplomityön tiivistelmä**

Tekijä:	Timo Ratilainen
Työn nimi:	Laadunvalvontakorttien visualisointityökalun toteutus laboratorioiden tiedonhallintajärjestelmään
Päivämäärä:	14.10.2004 Sivumäärä: 89
Osasto:	Tietotekniikka
Professuuri:	Tik-76 Ohjelmistoliiketoiminta ja -tuotanto
Työn valvoja:	Professori Reijo Sulonen
Työn ohjaaja:	DI Raimo Oinas

Tiivistelmäteksti:

Nykyaikaisessa kemian teollisuuden laboratoriotyöskentelyssä asetetaan suuria vaatimuksia prosessien ja tulosten laadulle. Tämän vuoksi laadunvalvonta on oleellinen osa laboratorioiden jokapäiväistä työtä. Laadunvalvontamenetelmistä erityisesti laadunvalvontakortteja käytetään paljon mittausprosesseista saatavien tulosten laadunvalvontaan. Yleensä normaalien näytteiden joukkoon asetetaan tunnettuja referenssinäytteitä, joiden tuloksista piirretään muuttujakohtaisesti laadunvalvontakortteja. Perinteisesti laadunvalvontakortteja on piirretty käsin millimetripaperille tai käytetty taulukkolaskentaohjelmaa apuna. Laboratorioiden tiedonhallintajärjestelmien yleistyttyä myös tarve automatisoidulle laadunvalvonnalle on kasvanut. Usein automatisoinnilla tavoitellaan myös lisää tehokkuutta ja tarkkuutta laadunvalvontaprosessiin.

Tämän diplomityön teoriaosuudessa esitellään laadunvalvontakorttien teoriaa ja käyttöä kemian laboratorioissa. Diplomityön toteutusosassa käydään läpi projekti, jonka lopputuloksena oli Java-pohjainen laadunvalvontakorttien visualisointisovelma. Visualisointisovelma myös tuotteistettiin onnistuneesti ja toimitettiin käyttöön useille eri asiakkaille Pohjoismaissa.

Avainsanat:
laadunvalvonta, laadunvalvontakortit, laadunvalvontakomponentti, visualisointi, LIMS-järjestelmät, laboratorioiden tiedonhallintajärjestelmät, Java, sovelma, SPC, tilastollinen prosessin ohjaus

Helsinki University of Technology **Abstract of the Master's Thesis**

Author:	Timo Ratilainen
Name of the Thesis:	Implementing control charts' visualization tool for laboratory information management systems
Date:	October 14, 2004 Pages: 89
Department:	Computer Science and Engineering
Professorship:	Tik-76 Software Business and Engineering
Supervisor:	Professor Reijo Sulonen
Instructor:	M.Sc. Raimo Oinas

Abstract:

Modern chemical industry's laboratory work sets great demands on the quality of the processes and results. Hence, quality control is an essential part of laboratories' every day work. From all of the quality control methods especially the quality control charts are used a great deal to control the quality of the measuring process's results. Usually known reference samples are placed among the normal samples, and they are used for drawing quality control charts according to the variables. Traditionally, the quality control charts have been drawn by hand on graph paper, or by using a spreadsheet program. Since the laboratory data management systems have become more common, there has also been an increase in the need for automatic quality control. The automatization is also often an attempt to increase the efficiency and accuracy of the quality control process.

The theoretical part of this thesis introduces the theory and usage of quality control charts in chemistry laboratories. The practical part of the thesis deals with a project that results in a Java-based quality control chart visualization applet. The applet was also successfully produced and delivered to several clients in the Nordic countries.

Keywords:

quality control, Shewhart's control charts, quality control module, visualization, LIMS-systems, laboratory information management system, Java, applet, SPC, statistical process control

Alkulause

Tämä työ on tehty Software Point Oy:ssä ja työn ohjaajana on toiminut diplomi-insinööri Raimo Oinas. Työn valvojana ja tarkistajana on toiminut professori Reijo Sulonen.

Esitän kiitokseni kannustuksesta ja ohjauksesta Samille, Tomille, Karolle, Juhalle, Marjutille, Jykälle, Tuijalle, Jukalle, Annille, Larille, Tancredille ja Markukselle sekä muille Software Pointin työntekijöille, jotka auttoivat tämän työn kanssa. Erityisesti kiitän Andreaa, Jussia ja Raimoa, jotka suuresti auttoivat tämän työn ideoimisessa ja toteutuksessa. Suuret kiitokset kuuluvat myös mukana olleille asiakkaille. Lisäksi kiitän professori Reijo Sulosta kannustavista ja erittäin asiantuntevista ohjeista, joilla tämäkin työ saavutti kypsyyspisteen.

Kiitän myös filosian maisteri Antti Tikkasta työn kieliasun tarkistamisesta.

Espoossa 14.10.2004

Timo Ratilainen

Timo Ratilainen

Laadunvalvontakorttien visualisointityökalun toteutus laboratorioiden tiedonhallintajärjestelmään

Sisällysluettelo

TERMIT

Alempi huomautusraja	Katso *Valvontakortin rajat.*
Alempi hälytysraja	Katso *Valvontakortin rajat.*
Alempi valvontaraja	Katso *Valvontakortin rajat.*
Applet	Katso *Sovelma.*
Aritmeettinen keskiarvo	(arithmetic mean, arithmetic average, μ, $\overline{X}$, $\overline{Y}$) Lukujen summa jaettuna niiden lukumäärällä. Keskiarvon käyttö edellyttää, että mitattava muuttuja on välimatka- tai suhdeasteikollinen, kuten esimerkiksi pituus ja ikä. Keskiarvomuuttujaksi eivät esimerkiksi sovi sukupuoli ja koulutus muuttujat. Keskiarvoa voidaan käyttää, kun jakauma on yksihuippuinen ja symmetrinen keskiarvon suhteen. Keskiarvo ei ole sopiva keskiluku kuvaamaan muuttujia, joissa on selvästi poikkeavia havaintoja, koska keskiarvo on herkkä poikkeaville arvoille.
AWT	(Abstract Windows Toolkit) Vanha Java-luokkakirjaston mukana tullut käyttöliittymäkirjasto.
EAServer	Sybasen kehittämä avoin sovelluspalvelin, jota myös LabVantage Sapphire –ohjelmisto käyttää.
Hajontaluku	Hajontaluvut kuvaavat kuinka voimakkaasti havaintoarvot ovat keskittyneet tai hajaantuneet jakauman keskikohtaa kuvaavan tunnusluvun ympärille. Pieni hajonta ilmaisee havaintoyksiköiden poikkeavan vain vähän jakauman keskiluvusta. Suuri hajonta kertoo, että havaintoaineistossa on paljon vaihtelua keskiluvun ympärillä. Hajontalukujen avulla ilmaistaan siis havaintoaineistossa esiintyvän vaihtelun määrää. Käytetyimpiä hajontalukuja ovat keskihajonta, varianssi ja vaihteluväli.

Jaguar	Katso *EAServer*.
JAR	(Java ARchive) Tiedostomuoto, jossa useita Java-luokkia ja muita resursseja on pakattu yhteen tiedostoon.
Java	Tuoteperhe, joka on kehitetty erityisesti tietoliikenneverkkoja ja alustariippumattomuutta varten. Yleensä Javalla tarkoitetaan tuoteperheen laajasti käytössä olevaa ohjelmointikieltä.
jFreeChart	LGPL-lisenssin (GNU Lesser General Public Licence) alla julkaistu Swing-pohjainen kuvaajien piirtokomponentti.
JRE	(Java Run Time Environment) Javan ajonaikainen ympäristö.
Keskiarvo	Katso *Aritmeettinen keskiarvo*.
Keskihajonta	(Standard deviation, s) Keskihajonta kuvaa havaintoarvojen keskimääräistä etäisyyttä keskiarvosta.
Keskiluku	Keskiluvut kuvaavat jakauman keskikohtaa ja ilmoittavat muuttujan keskimääräisen, tyypillisen tai yleisimmän arvon. Yleisimpiä keskilukuja ovat aritmeettinen keskiarvo, mediaani ja moodi.
Keskiviiva	Katso *Valvontakortin rajat*.
Kohdearvo	(target value, target line, target limit) Kohdearvolla tarkoitetaan valvontakortin arvoa, joka olisi optimaalinen laadunvalvontatulos. Katso myös *Valvontakortin rajat*.
LAL	Lower Alarm Limit, katso *Valvontakortin rajat*.
LCL	Lower Control Limit, katso *Valvontakortin rajat*.
LGPL	

LIMS	(Laboratory Information Management System) Laboratorion tiedonhallintajärjestelmä. Järjestelmällä voidaan hallita laboratorion tietovarastoja, kuten näytteitä, tuloksia, raportteja ja muita resursseja.
LNL	Lower Notification Limit, katso *Valvontakortin rajat.*
MVC	(Model-View-Controller) Suunnittelumalli, jossa esitysmuoto erotetaan sisällöstä. Yksinkertaistaen Swing-käyttöliittymäkirjastossa tämä tarkoittaa, että käyttöliittymäkomponentin ulkonäkö on tallennettu yhteen olioon ja sisältö toiseen.
OOA	(Object Oriented Analysis) Oliopohjainen analyysi.
OOD	(Object Oriented Design) Oliopohjainen suunnittelu.
Oracle	Yleisesti käytetty tietokantajärjestelmä.
QC-moduuli	Katso *QC-sovelma.*
QC-projekti	Katso *QC-sovelma.*
QC-sovelma	QC-projektin lopputuotteena syntyvä Java-pohjainen sovelma, jota voidaan käyttää osana Sapphire-järjestelmän WWW-käyttöliittymää.
Software Point Oy	Virallisesti Whitelake Software Point Oy, yritys joka antoi mahdollisuuden tämän diplomityön tekoon. Suomessa yrityksestä käytetään nimeä Software Point Oy. Yritys on erikoistunut laboratorioiden tiedonhallintajärjelmien toimittamiseen.
Sovelma	(Applet) Java-pohjainen ohjelma, joka ajetaan asiakaskoneella, yleensä selaimessa. Asiakaskoneessa tulee olla Javan ajonaikainen ympäristö (JRE) asennettuna.
SPC	(Statistical Process Control) Tilastollinen prosessin ohjaus.
Swing	Java-pohjainen käyttöliittymäkirjasto, jolla voidaan

toteuttaa monipuolisia käyttöliittymiä.

UAL — Upper Alarm Limit, katso *Valvontakortin rajat.*

UCL — Upper Control Limit, katso *Valvontakortin rajat.*

UNL — Upper Notification Limit, katso *Valvontakortin rajat.*

Valvontakortin rajat — Valvontakortin rajoilla tarkoitetaan kortille laskettuja ja yleensä myös piirrettyjä rajoja, joiden perusteella voidaan päätellä, onko kortilla poikkeamia laadun suhteen vai ei. Rajoina käytetään yleensä vakiovälein kohdearvoista poikkeavia lukuja. Yleensä välit lasketaan keskihajonnan monikertoina, esimerkiksi:

Ylempi hälytysraja (UAL) = kohdearvo + 3 x keskihajonta
Ylempi valvontaraja (UCL) = kohdearvo + 2 x keskihajonta
Ylempi huomautusraja (UNL) = kohdearvo + 1 x keskihajonta
Alempi huomautusraja (LNL) = kohdearvo - 1 x keskihajonta
Alempi valvontaraja (LCL) = kohdearvo - 2 x keskihajonta
Alempi hälytysraja (LAL) = kohdearvo - 3 x keskihajonta

Vakioina voidaan käyttää myös muita arvoja ja yhtälöt vaihtelevat korttityypeittäin. Myös rajojen nimet saattavat vaihdella. Yleensä laadunvalvontakorttiin piirretään vielä keskiviiva (kohdearvo), joka voi olla esimerkiksi kaikkien tulosten aritmeettinen keskiarvo.

Valvontakortti — Graafinen menetelmä prosessin tilan hallintaan kuvaajaan valittujen muuttujien kautta. Käytetään päätöksentekoon prosessin laadun osalta. Valvontakorttityyppejä on useita erilaisia, mutta yleisimmin käytetyt ovat $\overline{X}$ - ja R - laadunvalvontakortit.

Varianssi — Hajontaluku. Käytetään lähinnä teoreettisissa tarkasteluissa, koska varianssia on vaikea

	havainnollistaa. Keskihajonta on havainnollisempi, koska keskihajonnan yksikkö on sama kuin havaintojen. Esimerkiksi jos henkilöiden paino-muuttujan keskihajonta olisi 10 kg, olisi varianssi 100 kg^2.
Whitelake Software Point	Katso *Software Point Oy.*
WSP	Katso *Software Point Oy.*
Ylempi huomautusraja	Katso *Valvontakortin rajat.*
Ylempi hälytysraja	Katso *Valvontakortin rajat.*
Ylempi valvontaraja	Katso *Valvontakortin rajat.*
WWW	(World Wide Web) kaikki Internetissä olevat resurssit ja käyttäjät, jotka käyttävät HTTP-protokollaa (Hyper Text Transfer Protocol). Internetin kautta saatavissa oleva tieto.

JOHDANTO

Software Point Oy on laboratorioiden tiedonhallintaan ja toiminnanohjaukseen keskittynyt yritys. Software Point on pohjoismaiden johtava laboratorioiden tiedonhallinnan, tuotannon ja tutkimuksen laadunvarmistuksen ohjelmistoja ja vahvaa tietoteknistä asiantuntemusta tarjoava yritys. Näillä osa-alueilla Software Pointilla on merkittävä markkina-asema. Software Pointin asiakkaina on yhteensä satoja yksityisen ja julkisen puolen kemiallisia laboratoriota pohjoismaissa ja joissakin Keski- ja Etelä-Euroopan maissa. Tämän työn toteutusosa on tehty Software Pointille.

Yhä paisuvassa mittalaitteiden ja analyysimenetelmien kirjossa kemiallisilla laboratorioilla on kasvava tarve valvoa, ylläpitää ja kehittää analyysiprosessiensa laatua. Analyysiprosessien laatu sisältää muun muassa prosessista saatavien tulosten luotettavuuden ja tarkkuuden seuraamisen. Analyysit tehdään usein automaattisilla laitteilla, jotka voivat tutkia satoja näytteitä samalla kertaa, jolloin lopullisia tuloksia voi tulla tuhansia kappaleita lyhyessäkin ajassa. Usein kemiallisten laboratorioiden analyysimenetelmät ovatkin prosessinomaisia, jolloin niihin voidaan prosessien toistettavuudesta johtuen luontevasti soveltaa tilastollisia menetelmiä. Tällöin viime vuosisadan alkupuolella kehitetty *tilastollinen prosessin ohjaus* on hyvä valinta laadun valvontaan, ylläpitoon ja kehitykseen. Tilastollinen prosessin ohjaus sisältää suuren joukon erilaisia menetelmiä, mutta tässä työssä keskitytään erityisesti kemiallisten laboratorioiden käyttämiin laadunvalvontakortteihin. Laadunvalvontakortit antavat mahdollisuuden tehokkaaseen analyysiprosessien laadunvalvontaan, koska ne pystyvät paljastamaan erityisistä syistä (selvitettävissä olevat) johtuvat poikkeamat.

Nykyään useimmat kemialliset laboratoriot valvovat analyysiprosessiensa laatua laadunvalvontakorteilla, mutta menetelmät eivät ole yleensä kovin kehittyneitä, vaikka itse teoria on ollut olemassa jo lähes vuosisadan ajan. Usein laboratorioissa käytetään perinteistä millimetripaperia, johon piirretään halutut laadunvalvontakortit käsin. Toinen suosittu tapa on käyttää yksinkertaisia automatisoituja menetelmiä, esimerkiksi Microsoft Excel-taulukkolaskentaohjelmalla toteutettuja kuvaajia. Tällöin ei kuitenkaan saavuteta monia integroidun laadunvalvontatyökalun etuja,

kuten tulosten reaaliaikaista valvontaa tai monimutkaisten sääntökuvioiden automaattista tunnistamista.

Tämän diplomityön teoriavaiheessa on etsitty lisää tietoa ja ymmärrystä tilastollisesta prosessin ohjauksesta ja laadunvalvontakorteista kemiallisissa laboratorioissa. Asiakasvaatimusvaiheessa otettiin huomioon, se että eri asiakkailla saattaa olla hyvinkin erilaisia vaatimuksia ja tapoja laadunvalvonnassa, jolloin lähtökohtana oli, että kaikkia toiveita tuskin pystytään toteuttamaan. Vaatimusten keruuvaiheessa tarkoituksena oli saada kasaan sopiva paketti, joka tyydyttäisi useimpia asiakkaita ja jättäisi mahdollisuuden kehittää työkalua jatkossa asiakkaiden toivomusten mukaan.

Software Pointin toimittamaan LabVantage Sapphire LIMS-järjestelmään tarvittiin käyttäjäystävällinen työkalua, jolla voidaan tuottaa laadunvalvontakortteja. Tässä työssä tutkitaan visualisointityökalua varten tarvittavaa teoriapuolta ja käydään läpi varsinaisen laadunvalvontatyökalun toteutusprojekti. Visualisointityökalun toteutus tehtiin Software Pointin sisäisenä projektina, jolloin yksittäisellä asiakkaalla ei ollut koko päätäntävaltaa projektin lopputulokseen. Visualisointikomponentin vaatimukset kerättiin asiakastoiveina useilta eri asiakkailta. Vaatimusten keräämisprosessin yhteydessä syntyi kattava kuva kemiallisten laboratorioiden laadunvalvonnasta laadunvalvontakorttien osalta.

Toteutusprojektissa on otettu mahdollisuuksien mukaan huomioon tulevaisuuden laajennukset, esimerkiksi uusien korttityyppien tukeminen, monimuuttujakorttien tukeminen ja visualisointityökalun muuttaminen siten, että se toimisi myös muissa tiedonhallintajärjestelmissä kuin pelkästään LabVantage Sapphire LIMS-järjestelmässä.

1 Taustaa

Tässä kappaleessa käydään läpi syitä, joiden vuoksi laadunvarmistuskomponenttia lähdettiin toteuttamaan. Lisäksi esitellään tärkeimmät vaatimukset toteutettavalle laadunvarmistuskomponentille ja käydään läpi kirjoittajan osuutta kokonaistyömäärästä.

1.1 Nykyiset järjestelmät

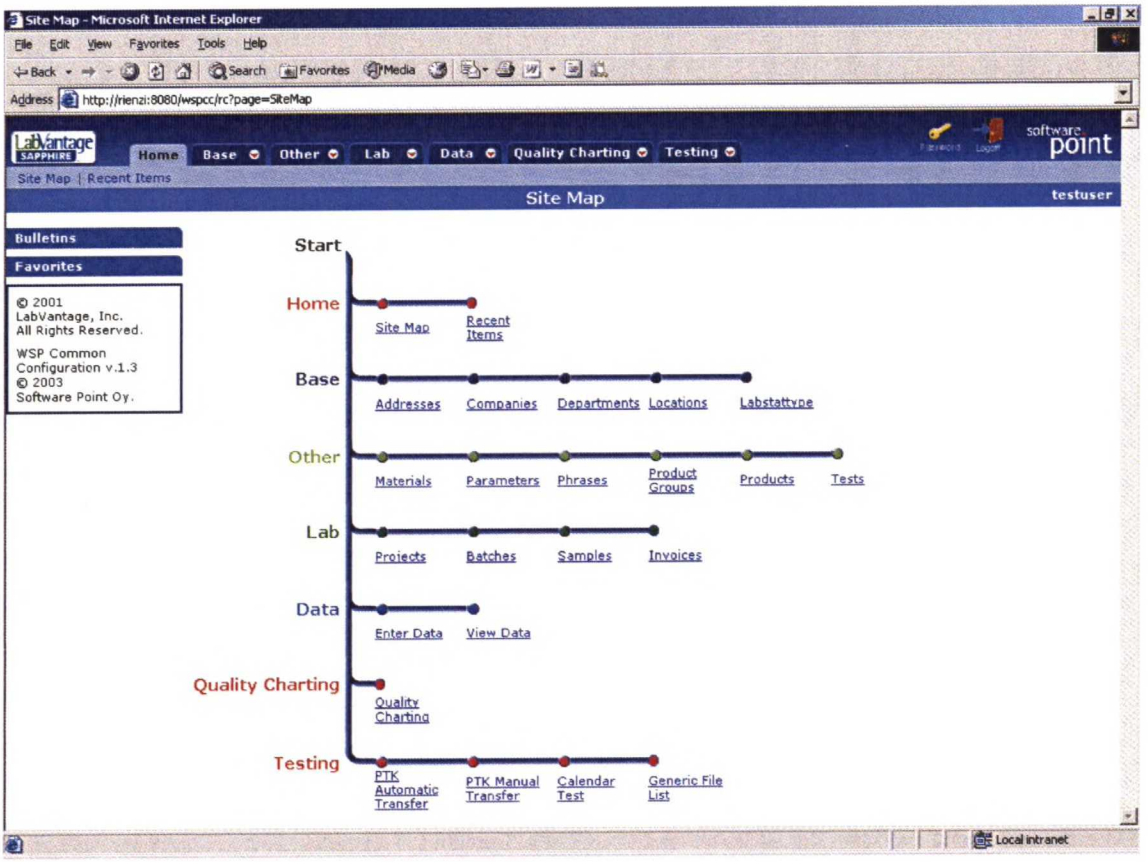
Software Point Oy on perustettu vuonna 1992 spin-offina biotekniikkaan erikoistuneesta Provivo Oy -yrityksestä. Provivon bioautomaatio-osastolla oli käynnissä Bioworkstation-ohjelmiston kehittämisohjelma, joka oli samalla osa kansallista biotekniikkaohjelmaa. Projektin valmistuessa vuonna 1992 Bioworkstationista tehtiin standardituote nimeltään BioBOSS ja bioautomaatio-osaston henkilöstöstä muodostettiin Software Point Oy, johon kaikki Provivon ohjelmistokehitys siirrettiin. [WSP]

Alkuvuosina Software Point keskittyi biotekniikan yrityksille suunnattuihin ohjelmistoihin, mutta aikojen kuluessa Software Pointista on kasvanut laboratorioiden koko tiedonhallinnan ja toiminnanohjauksen ohjelmistoja tarjoava yritys. Software Pointin tuotepaletti käsittääkin tällä hetkellä yli kymmenen ohjelmistotuotetta. Software Pointilla on merkittävää liiketoimintaa Suomen lisäksi Ruotsissa, Norjassa ja Tanskassa. Yritys työllistää noin 40 työntekijää.[WSP]

Laboratorion tiedonhallintajärjestelmät (LIMS) ovat laboratorioiden tietovarastoja, joista tärkeä tieto on helposti saatavissa. Järjestelmässä on yleensä tietoa näytteistä, tuloksista ja laboratorioiden resursseista, kuten laitteista, analyysimenetelmistä ja henkilöstöstä. LIMS-järjestelmää voidaan käyttää myös tiedon jatkojalostamiseen, raportointiin, laadunvalvontaan, työn ohjaamiseen ja automatisointiin. Yleensä ainakin osa laboratorion digitaalisista analyysilaitteista liitetään LIMS-järjestelmään, jolloin esimerkiksi analyysitulokset voidaan siirtää järjestelmänn automaattisesti. [Kuokkanen]

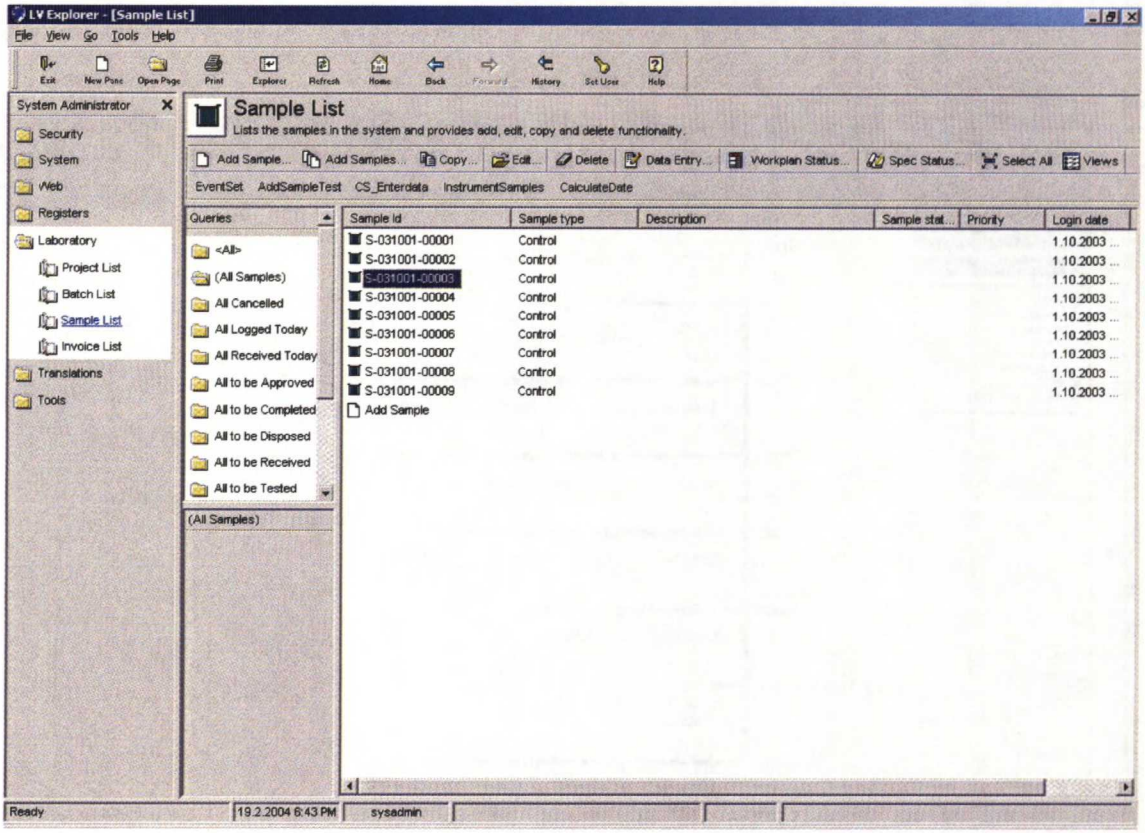
Nykyään yksi merkittävimmistä tuotteista Software Pointin tuotepaletissa on LabVantage Sapphire –ohjelmisto. LabVantage Sapphire -ohjelmisto on Yhdysvaltalaisen LabVantage Solutions, Inc. -yrityksen kehittämä LIMS-järjestelmä. Software Point solmi yhteistyösopimuksen LabVantagen kanssa vuonna 2000 ja tähän sopimukseen perustuen Software Point voi yksinoikeudella toimittaa Sapphire-järjestelmää Pohjoismaissa.

Sapphire-järjestelmä toimii sekä Windows- että WWW-pohjaisena ohjelmistona. Asiakkaille toimitetaan tarvittaessa vain toinen tai molemmat, riippuen asiakkaan tarpeista ja käyttöympäristöstä. Käyttöliittymän lisäksi tarvitaan erillinen palvelinohjelmisto (EAServer) ja Microsoft SQL Server- tai Oracle –tietokanta. Varsinaisten lisenssin myymisen lisäksi tarvitaan yleensä vielä konfigurointityötä, jossa esimerkiksi WWW-pohjainen Sapphire-järjestelmä räätälöidään vastaamaan asiakkaan työskentely-ympäristöä. Räätälöinnissä Sapphire-järjestelmä yleensä myös linkitetään laboratorioiden analyysilaitteisiin. Alla olevassa kuvassa (Kuva 1) on esimerkkikonfiguraatio Sapphire-järjestelmän pääsivusta, josta käyttäjä voi aloittaa haluamansa toiminnon. Kuvassa oleva Sapphire-konfiguraatio on rakennettu pelkästään testikäyttöön.



Kuva 1: Esimerkki LabVantage Sapphire -järjestelmän WWW-käyttöliittymän pääsivusta

Seuraavassa kuvassa (Kuva 2) on esitetty sama testiympäristö ajettuna Microsoft Windows -pohjaisesta LVX-käyttöliittymästä. LVX-käyttöliittymä on tarkoitettu Sapphire-järjestelmän Windows-pohjaiseen käyttöön. Kyseisessä kuvassa ollaan tutkimassa järjestelmään syötettyjä näytteitä.



Kuva 2: Esimerkki LabVantage Sapphire -järjestelmän LVX-käyttöliittymästä

1.2 Laadunvalvontakorttien tarve

Laadunvalvontakorttien visualisointia varten asiakkaat ovat yleensä ostaneet joko kokonaan uuden ohjelmiston, esimerkiksi Northwest Analytical Inc. -yrityksen Quality Analyst -ohjelmiston, tai sitten käyttäneet puoliautomatisoituja menetelmiä, kuten Excel-taulukkolaskentaohjelmaa. Usein laboratoriot kuitenkin kokevat Quality Analystin kaltaisen laajan tilasto-ohjelmiston käytön turhan raskaaksi. Kemiallisten laboratorioiden kannalta Quality Analystissä on paljon turhia toimintoja. Quality Analyst on yleispaketti, jossa on kaikkeen tilastolliseen laskentaan ja grafiikkaan liittyviä toimintoja sekä se on hankala integroida ja käyttää. Toisaalta, jotkut laboratoriot hoitavat laadunvalvontakorttien käsittelyn vielä paperimuodossa.

Tässä työssä tutustutaan laadunvalvontakorttien teoriaan ja hankitaan lisää tietoa siitä, kuinka asiakkaat todellisuudessa käyttävät laadunvalvontakortteja. Tämän jälkeen suunnitellaan ja toteutetaan laadunvalvontakomponentti LabVantage Sapphire -järjestelmään.

1.3 Kirjoittajan osuus

Tämän työn kirjoittajalla ei ole ollut henkilökohtaisia kokemuksia edellä mainituista laboratorioiden työtehtävistä ja ongelmista, koska hän ei ole työskennellyt kemiallisissa laboratorioissa, muuten kuin korkeakouluopintojen puolesta. Näin ollen oleellinen osa laadunvalvontakomponentin toteutusprojektista koostui keskustelusta asiakkaiden kanssa ja vaatimusten keräämisestä. Tässä työssä kirjoittaja oli aktiivinen, ja hän kävi kymmeniä keskusteluja niin varsinaisissa kokouksissa kuin myös sähköpostin ja puhelimen välityksellä. Kirjoittaja vieraili myös asiakkaiden laboratorioissa seuraamassa, kuinka laboratorioissa työskentelevät ihmiset laadunvalvontaa tämän työn aihepiirin puitteissa tekevät. Ilman näitä ponnisteluja olisi laadunvalvontakompontin toteutusvaihetta ollut turha aloittaa. Yhteistyötä tehtiin Rautaruukin, Valion, Yhtyneiden laboratorioiden, Kansanterveyslaitoksen, Geologian Tutkimuskeskuksen, Tullilaboratorion ja Central Soyan kanssa, josta heille suuret kiitokset.

Laadunvalvontakomponentin suunnittelu- ja toteutusvaiheet ovat lähes sataprosenttisesti kirjoittajan itsensä tekemiä. Kirjoittaja sai kuitenkin apua useilta henkilöiltä sekä Software Pointin sisältä että asiakkaiden organisaatioista.

Jussi Palssa on auttanut merkittävästi tietokantariippuvien osioiden suunnittelussa, kuten myös laadunvalvontakomponentin tietokantamallin suunnittelussa ja toteutuksessa. Yrityksen toimitusjohtaja, tekniikan tohtori Andrea Holmberg, on antanut useita hyödyllisiä vinkkejä siihen, mitä kannattaa ja mitä ei kannata toteutttaa. Esimieheni Raimo Oinas puolestaan auttoi tiedoillaan Sapphire-järjestelmästä. Andrealla ja Raimolla on kymmenien vuosien kokemus Suomen laboratoriokentästä tiedonhallintajärjestelmien näkökulmasta. Lisäksi Anna Laukkanen auttoi etsimällä sopivan käyttöliittymäkirjaston projektin käyttöön (liite A ja B) ja Tancred Lindholm auttoi etsimällä jar-pakettien supistustyökalun.

2 Tilastollinen prosessin ohjaus

Laadun valvontaan ja kehitykseen on olemassa suuri joukko erilaisia menetelmiä, kuten tilastollinen prosessin ohjaus (Statistical Process Control, SPC), Design of Experiment (DOE), Total Quality Management (TQM), Company-Wide Quality Control (CWQC), Total Quality Assurance (TQA), Just-In-Time (JIT), Lean Manufacturing, Poka-Yoke, Reengineering, Theory of Constrains, Agile Manufacturing ja Six-Sigma [Montgomery, Griffith]. Tässä työssä keskitytään kuitenkin vain ensiksi mainittuun, tilastolliseen prosessin ohjaukseen, koska se on laajasti käytössä juuri kemiallisissa laboratorioissa.

2.1 Johdanto SPC-menetelmään

Tilastollinen prosessin ohjaus on käytännössä joukko matemaattis-visuaalisia menetelmiä, joiden antamia tuloksia analysoimalla voidaan prosessia valvoa ja ohjata [Griffith]. Vaikka määritelmä onkin melko yksiselitteinen, käytetään tilastollisesta prosessin ohjauksesta suomessa useita eri termejä, kuten SPC-työkalu, SPC-tekniikka ja SPC-menetelmä. Myös auki puretuista lyhenteistä on useita eri versioita, kuten tilastollinen laadunvalvonta ja tilastollinen prosessin ohjaus. Tässä työssä käytetään termejä SPC-menetelmä ja tilastollinen prosessin ohjaus.

Vaikkakin matemaattis-visuaaliset työkalut ovatkin SPC-menetelmän oleellinen osa, menetelmän tarkoitus on myös rakentaa ympäristö, missä koko organisaation henkilöstö ottaa kaikissa toimissaan huomioon laadunparannustavoitteet. SPC-menetelmän mukaan tämä on parhaiten saavutettavissa, kun organisaation johto on tietoisesti mukana laadunparannusprosessissa [Montgomery, Griffith]. Tässä työssä keskitytään kuitenkin pää-asiallisesti SPC-menetelmän matemaattis-visuaalisiin työkaluihin.

Yleensä SPC-menetelmän matemaattis-visuaalisiin työkaluihin lasketaan kuuluvaksi seuraavat seitsemän työkalua (*the magnificent seven*): [Montgomery]

- Histogrammi (histogram)

- Tarkistuslista (check sheet)
- Pareto-kortti (pareto chart)
- Syy-seuraus kuvaaja (cause-and-effect diagram)
- Virhetiheys kuvaaja (defect concentration diagram)
- Hajontakuvaaja (scatter diagram)
- Valvontakortti (control chart)

SPC-menetelmässä on oleellisena osana prosessi, jonka toimintoihin tilastollisia menetelmiä kohdistetaan. Prosessi-termille löytyy useita erilaisia määritelmiä, mutta tämän työn kemiallis-tietotekniseen aihepiiriin sopiva määritelmä prosessi-termille on:

> "Prosessilla tarkoitetaan toimintosarjaa, joka jalostaa tuotetta tai palvelua. Prosessilla on tarkasti määritelty alku ja tarkasti määritelty loppu. On määriteltävä selvästi, mitkä ovat prosessin tarvitsemat panokset, mitä sen tulee tuottaa, kuinka sen tulee tapahtua ja minkälaisia ovat odotetut tulokset. Kaikilla prosesseilla on asiakas – sisäinen tai ulkoinen. Ominaista prosesseille on myöskin toistettavuus, eli ne viedään läpi useita kertoja." (Suomen laatukeskus, 1998)

Prosessin määritelmä on siis hyvin määräävä: tulee tietää mitä tehdään, kuinka se tehdään ja mitä on tuloksena. Prosessin toistuva rakenne on tärkeä edellytys SPC-menetelmän tilastollisten menetelmien käytölle. Tilastollisilla menetelmillä on harvoin käytännön merkitystä, jos käytössä on vain muutama havainto prosessista.

Prosessin määritelmän pohjalta ajatellen ei ole ihme, että teollisen yhteiskunnan lyhyehkössä historiassa laatuajattelun peruskivet on muurattu jo viime vuosisadan alkupuolella. Tämä tapahtui samoihin aikoihin, kun prosessiajattelu tuli suuressa mittakaavassa tärkeäksi. Vuosisadan alkupuolella vaikuttanut amerikkalainen Walter A. Shewhart työskenteli Bellin puhelinlaboratoriossa ja puhelintehtaassa, jossa puhelinkojeiden valmistus oli prosessiluonteista. Hän huomasi, että pyrkimys korjata kaikki tuotteissa havaitut virheet johtaa lopulta kaaoksen lisääntymiseen ja

näin ollen mahdottomaan tilanteeseen koko prosessia ajatellen. Shewhart totesi, että vain pienelle osalle virheistä oli jokin erityinen syy. Pääosa virheistä oli satunnaisia. Näin hän päätteli, että ainoa mahdollisuus vaikuttaa virhemäärään, oli vaikuttaa prosessiin erityissyiden kautta. Tämän vuoksi Shewhart kehitti säännöstön, jonka avulla virheet voitiin jakaa satunnais- ja erityissyihin (luonnollinen syy ja poikkeava häiriö). Tämä havainto on tilastollisen prosessin ohjauksen kulmakivi vielä nykyäänkin. [Tuurala]

Shewhart julkaisi vuonna 1931 kirjan: "Economic Control of Quality of Manufactured Product" [Shewhart]. Kirja käsittelee laadunvalvontaa tilastollisten menetelmien näkökulmasta, graafista visualisointia painottaen, ja oli pitkään pohjana laadunvalvonnan tieteelliselle kehittämiselle. Huomattava osa nykyisin käytössä olevista laatumenetelmistä on seurausta Shewhartin ideoista ja Bellin laadunvalvontaosastolla tuolloin kehitetyistä menetelmistä. [Tuurala]

Tulee kuitenkin ottaa huomioon, että laadunvalvontakortit on alun perin kehitetty tuotannollisiin prosesseihin, kuten autonvalmistukseen, ja siellä käytetyt laadunvalvontakortit eivät ole suoraan käyttökelpoisia kemiallisten laboratorioiden analyysiprosesseissa.

2.2 Laadunvalvontakortit

SPC-menetelmän eniten käytetty työkalu on valvontakortti (quality control chart, statistical control chart). Myös laadunvalvontakomponentin toteutusprojektin kannalta laadunvalvontakortit ovat avainasemassa, joten tulevissa kappaleissa käydään yksityiskohtaisesti läpi laadunvalvontakorttien tekniikka. Valvontakortin idean esitti ensimmäisen kerran Shewhart työskennellessään Bellin puhelinlaboratoriossa 1920-luvun alkupuolella. Shewhartin mukaan laadunvalvontakorteilla voidaan eliminoida epänormaali vaihtelu prosessissa erottamalla selvitettävissä olevista syistä ja sattumanvaraisista syistä johtuvat vaihtelut toisistaan. SPC-menetelmässä pyritään ohjaamaan prosessia niin, että siihen vaikuttavat ainoastaan yleiset hajontaa aiheuttavat tekijät. Laadunvalvontakorteilla on kyky visualisoida poikkeamat ihmiselle helpommin ymmärrettävään muotoon. [Wheeler, Prichard, Wise, Griffith]

Laadunvalvontakorttien käyttö kemiallisissa laboratorioissa tapahtuu yleensä seuraavasti: [Asiakaskeskustelut]

1. Määritetään tarkkailtava muuttuja prosessista. Yleensä tämä on itsestään selvää, jos esimerkiksi määritetään alkuainepitoisuutta näytteestä.

2. Kerätään historiatietoa, joiden pohjalta määritetään valvontarajat, joiden sisällä tulevien tulosten tulisi pysyä. Historiatiedot kerätään yleensä referenssinäytteistä, joiden tulokset tiedetään etukäteen. Jos referenssinäytteitä ei voida käyttää, voidaan valvontarajat vaihtoehtoisesti laskea itse tuloksista.

3. Tarkkaillaan pysyvätkö uudet tulokset valvontarajojen sisäpuolella. Tässä käytetään apuna myös valvontasääntöjoukkoa (säännöistä enemmän kappaleessa *2.3*)

4. Jossain vaiheessa laadunvalvontakortti vanhenee, ja luodaan uusi kortti. Kortti voi vanheta esimerkiksi referenssinäyte-erän loppuessa. Tällöin voidaan hetken ajan kerätä tuloksia rinnakkaisiin valvontakortteihin ja palataan kohtaan kaksi.

Edellä mainittu vaiheistus saattaa poiketa laboratoriokohtaisesti.

Valvontakorttien tehokkuuden perusoletus on, että tutkittava prosessi täyttää kontrolloidun variaation (controlled variation) vaatimuksen historiatiedon keruun aikana. Kontrolloitu variaatio tarkoittaa, että prosessissa esiintyy vain satunnaisia poikkeamia. Historiatiedon keruussa lasketaan valvontakortin keskiarvo ja keskihajonta, joiden perusteella määritetään myös valvontarajat. Jos alkuvaiheen historiatieto ei ole kerätty kontrolloidusta prosessista, tulevat tulokset eivät voi olla historiatietojen mukaan määriteltyjen rajojen sisäpuolella muuten kuin satunnaisesti. Menetelmä siis muistuttaa matemaattista ekstrapolointia. [Wheeler92]

Valvontakortin rajojen määrittämisen jälkeen tarkkaillaan, noudattaako prosessi historiatiedoista saatua linjaa. Jos tulokset pysyvät rajojen sisäpuolella, saattaa prosessi olla hallinnassa. Jos tulevat tulokset eivät pysy rajojen sisäpuolella, voidaan olla varmoja, että prosessi on epävakaa ja korjaavat toimet voidaan aloittaa. [Wheeler92] Korjaavien toimien läpikäynti ei kuulu tämän työn aihepiiriin, mutta hyvä lähde aiheeseen on Asakan ja Ozekin kirja: Handbook of quality control tools. [Asaka]

Shewhartin mukaan prosessi on hallinnassa (process is in control, process is predictable), jos tunnetun historian pohjalta voimme ennustaa prosessin tulevaa käyttäytymistä. Tällöin ainoastaan satunnaiset syyt aiheuttavat vaihtelua ja prosessi tuottaa lopputuotetta, joka on lähempänä tavoitearvoja. [Wheeler92, Montgomery, Shewhart]

Laadunvalvontakorteista on olemassa useita tyyppejä, joita harvoin tarvitaan kaikkia saman prosessin valvonnassa. Yleensä voidaan valita prosessiin parhaiten sopivat korttityypit kirjallisuuden perusteella (liite C) tai kokeilemalla. Tämän työn aihepiirin liittyen tarkastellaan kahta yleisesti käytössä olevaa tyyppiä:

- X -kortit, joilla tarkastellaan rinnakkaisten tulosten keskiarvon vaihtelua

- R -kortit, joilla tarkastellaan rinnakkaisten tulosten pienimmän ja suurimman arvon erotuksen vaihtelua

Laadunvalvontakortit kehitettiin alun perin tuotantoprosesseihin, joissa niitä käytetään vielä nykyäänkin eniten. Kemiallisissa laboratorioissa prosessit ovat yleensä kuitenkin jonkin verran erilaisia. Kemiallisissa laboratorioissa laadunvalvontakorteilla pyritään lähinnä varmistamaan, että kemialliset menetelmät antavat luotettavia tuloksia. Varmistaminen tapahtuu yleensä siten, että prosessin läpi ajetaan ennalta tunnettuja referenssinäyteitä. Näin ollen mitattavat näytteet ovat niin samanlaisia, että mitattujen tulosten välinen hajonta voidaan unohtaa. [Asaka]

Kemiallisissa laboratorioissa laadunvalvontakorteilla on seuraavia tehtäviä:

- Valvoa ja ylläpitää tarvittavaa laatua. Tämä tapahtuu kirjaamalla laadunvalvontatulokset laadunvalvontakorttiin, josta nähdään tilanteet joissa jokin tulos rikkoo valvontarajoja. Jos rajoja rikotaan, voidaan aloittaa tarvittavat korjaavat toimenpiteet.

- Laadun dokumentointi myöhempää tarkastelua varten. Näitä tarkasteluja voivat aiheuttaa esimerkiksi takuu ja lakisääteiset tilanteet, tai organisaation sisäisen laadun katselmoinnit.

2.2.1 X -tyypin laadunvalvontakortti

X -tyypin (average chart) laadunvalvontakortissa tarkkaillaan tulosjoukon rinnakkaisten tulosten aritmeettista keskiarvoa. Rinnakkaisilla tuloksilla tarkoitetaan tuloksia, joista voidaan laskea lopullinen tulos, joka laadunvalvontakortilla näytetään. Rinnakkaisiksi tuloksiksi voidaan esimerkiksi ottaa samasta näytteestä samalla hetkellä saadut tulokset. X -tyypin laadunvalvontakortilla pyritään valvomaan prosessin keskimääräistä laatua. [Montgomery]

X -tyypin kortit voidaan jakaa X - ja $\overline{X}$ -tyypin kortteihin. X -tyypin kortilla ei ole rinnakkaisia tuloksia, kun taas $\overline{X}$ -tyypin kortilla niitä on. Kortit ovat ominaisuuksiltaan hyvin samankaltaiset ja laskennassa käytettävät matemaattiset yhtälöt ottavat automaattisesti huomioon rinnakkaisten tulosten lukumäärän. Tässä työssä käytetään vain X -tyypin korttimerkintää tarkoittamaan molempia korttityyppejä, jos ei muuta erikseen mainita.

X -tyypin laadunvalvontakortin soveltaminen tapahtuu seuraavasti. Ensin lasketaan tulosjoukon rinnakkaisten tulosten aritmeettinen keskiarvo.

$$\overline{X}_i = \frac{\sum_{i=0}^{k} X_i}{k}$$

Yhtälö 1 Rinnakkaisten tulosten aritmeettinen keskiarvo

Yhtälössä (Yhtälö 1) $\overline{X}_i$ tarkoittaa tulosjoukon tuloksen i arvoa, eli k:n rinnakkaisen pisteen aritmeettista keskiarvoa. Lisäksi X-tyypin laadunvalvontakortteihin merkitään keskiviiva, joka on edellä mainittujen rinnakkaisten tulosten aritmeettisten keskiarvojen aritmeettinen keskiarvo.

$$\overline{\overline{X}} = \frac{\sum_{i=0}^{K} \overline{X}_i}{K}$$

Yhtälö 2 Keskiviivan laskenta

Yhtälössä (Yhtälö 2) $\overline{\overline{X}}$ tarkoittaa kaikkien rinnakkaisten tulosten aritmeettisten keskiarvojen (K kappaletta) aritmeettista keskiarvoa. Arvoa $\overline{\overline{X}}$ sanotaan myös kohdearvoksi, koska se on usein laadunvalvontatulosten optimaalinen arvo, eli tulosten tulisi olla mahdollisimman lähellä kohdearvoa.

Seuraavaksi lasketaan keskihajonta.

$$s = \sqrt{\frac{\sum_{i=0}^{n} \left(\overline{X}_i - \overline{\overline{X}}\right)^2}{n-1}}$$

Yhtälö 3 Keskiarvojen keskiarvon keskihajonta

Yhtälössä (Yhtälö 3) s tarkoittaa keskihajontaa ja n tarkoittaa aikaisemmin lasketun keskiarvojoukon $\overline{X}$ alkioiden lukumäärää. [Wheeler92, Liite D, kappale *Mathematics*]

Tämän jälkeen laadunvalvontakorttiin lisätään joukko rajoja, joilla tarkkaillaan valitun muuttujan käyttäytymistä. Valvontarajat lasketaan suhteessa kohdearvoon $\overline{\overline{X}}$ käyttäen apuna keskihajontaa.

$$valvontaraja = \overline{\overline{X}} + \frac{as}{\sqrt{r}}$$

Yhtälö 4 Valvontarajat keskiarvojen keskiarvoon nähden

Yhtälössä (Yhtälö 4) s tarkoittaa keskiarvojen keskihajontaa ja $\overline{\overline{X}}$ tarkoittaa keskiarvojen keskiarvoa. Neliöjuuressa oleva r tarkoittaa rinnakkaisten tulosten lukumäärää. Vakio a voidaan valita vapaasti, mutta perustellusti. Mitä enemmän rinnakkaisia tuloksia on, sitä pienemmäksi valvontarajojen erotus käy ja sitä tarkemmin prosessin laatua voidaan valvoa. Yhtälöllä voidaan laskea kaikki X -tyypin laadunvalvontakorttien valvontarajat muuttamalla vakiota a sopivasti. Yleisimmät käytetyt vakiot ovat ± 3.0 ja ± 2.0, mutta prosessikohtaisesti voidaan valita muitakin vakioita. Seuraavassa listassa on lueteltu yleisimmin käytettyjen $\overline{X}$ -tyypin laadunvalvontakorttien valvontarajat (lähteestä riippuen rajoja saatetaan nimetä toisin):

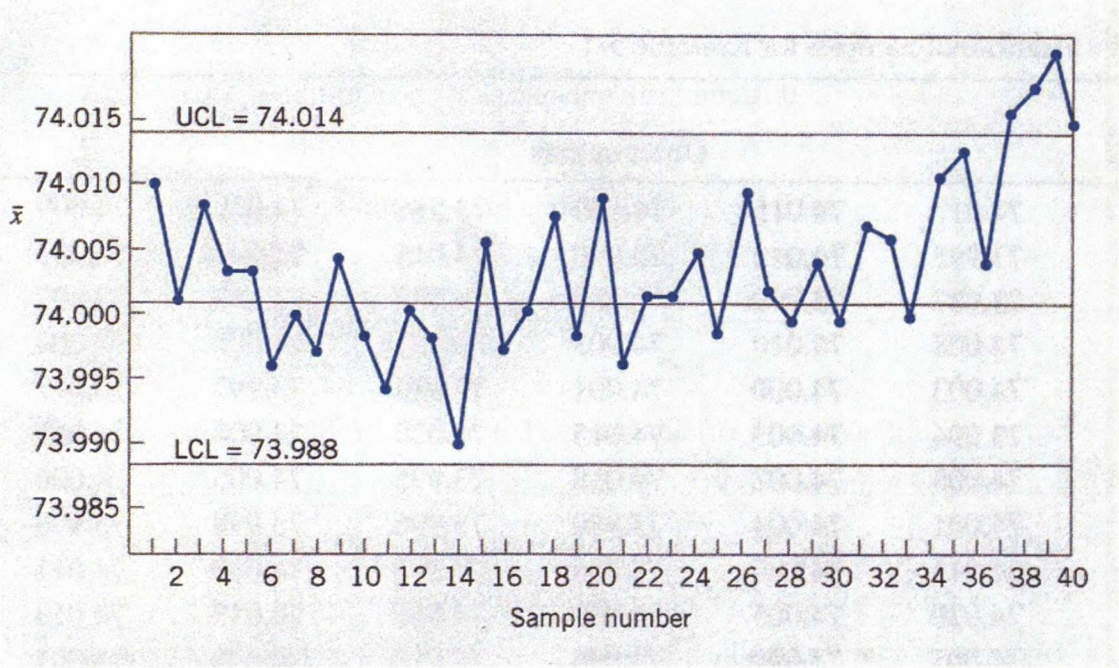
- +3.0 Ylempi valvontaraja (upper control limit, UCL)
- -3.0 Alempi valvontaraja (lower control limit, LCL)
- +2.0 Ylempi hälytysraja (upper warning limit, UWL)
- -2.0 Alempi hälytysraja (lower warning limit, LWL)

Esitetyt vakiot on todettu kirjallisuudessa kokeellisesti riittäviksi, koska niillä saavutetaan tarvittava herkkyys prosessin suhteen ilman, että vääriä hälytyksiä valvontarajojen rikkomisista tulisi liikaa. Laadunvalvontakorttien valvontarajat voidaan laskea esitetyillä vakioilla, vaikka tulosjoukko olisi hyvinkin vinoutunut. Vinoutuneella tulosjoukolla tarkoitetaan tulosjoukkoa, jonka tulokset eivät ole jakaantuneet pienimmän ja suurimman tuloksen välille tasaisesti. Jos prosessi on tilastollisesti hallittava, niin noin 99 – 100 prosenttia tulosjoukon arvoista on ± 3.0 rajojen sisäpuolella ja noin 90 – 98 prosenttia ± 2.0 rajojen sisäpuolella. Tämä tilastollinen tarkkuus on useimmiten riittävä. [Wheeler95]

Valvontarajojen määrittämisen jälkeen luodaan joukko sääntöjä, joiden mukaan havaintopisteitä verrataan valvontarajoihin. Havaintopisteellä tarkoitetaan laadunvalvontakortille piirrettävää pistettä. Havaintopiste voi olla suoraan prosessista saatu tulos, tai rinnakkaisista tuloksista saatu *laskennallinen tulos*. Jos

havaintopiste rikkoo jotain sääntöä, voidaan epäillä, että prosessissa vaatii tarkkailua ja mahdollisesti toimenpiteitä ongelman korjaamiseksi. Säännöistä on kerrottu enemmän kappaleessa 2.3.

Kuva 3 esittää yksinkertaista esimerkkitapausta $\overline{X}$ -tyypin laadunvalvontakortista. Korttiin on piirretty yksinkertaisuuden vuoksi vain ylempi ja alempi valvontaraja. Kortista havaitaan, että loppupään tulokset menevät valvontarajojen ulkopuolelle. Tästä voidaan tehdä arvio, että suunnilleen näytteen 37 kohdalla prosessissa on tapahtunut muutos, jota sitten voidaan lähteä jäljittämään itse prosessista. Saman tiedon havaitseminen ei-visuaalisesta taulukosta vaatisi enemmän aikaa ja olisi alttiimpi virheille.



Kuva 3 Yksinkertainen esimerkki X -kortista [Montgomery]

2.2.2 R -tyypin laadunvalvontakortti

R -tyypin (range chart) laaadunvalvontakortit ovat visuaaliselta olemukseltaan lähellä X -tyypin laadunvalvontakortteja, mutta näissä visualisoidaan rinnakkaisten tulosten suurimman ja pienimmän arvon eroja. R -tyypin laadunvalvontakorteilla valvotaan variaatiota yhden näytteen sisällä [Montgomery, Wise]. Yleensä X - ja R -tyypin laadunvalvontakortteja hyödynnetään samanaikaisesti

R -tyypin laadunvalvontakortin tulosten erotus lasketaan seuraavan yhtälön mukaan:

$$R_i^p = X_{\max}^p - X_{\min}^p$$

Yhtälö 5 Rinnakkaisten tulosten suurimman ja pienimmän arvon erotus

Yhtälössä (Yhtälö 5) p tarkoittaa rinnakkaisten tulosten joukkoa näytteelle i ja R_i^p tarkoittaa joukosta p laskettavaa suurimman $X_{\max}^p$ ja pienimmän $X_{\min}^p$ tuloksen erotusta näytteelle i.

Seuraavaksi R -tyypin laadunvalvontakortteihin lasketaan valvontarajat, kuten X -tyypin laadunvalvontakortteihin, tosin vain nollatason yläpuolelle (Yhtälö 6). R -tyypin laadunvalvontakorteissa ei lasketa rinnakkaisten tulosten keskiarvoa, vaan rinnakkaisten tulosten suurimman ja pienimmän arvon erotus, jolloin vaikuttavien tulosten joukko on pienempi.

$$valvontaraja = as$$

Yhtälö 6 Valvontarajat nollatasoon nähden

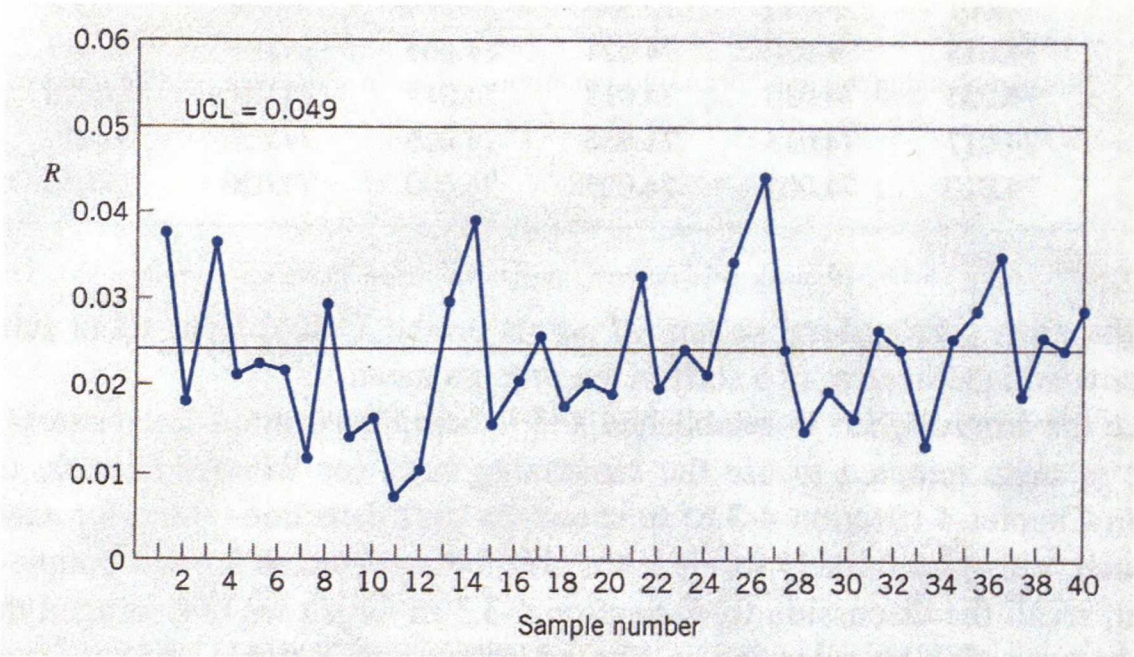
Tämän jälkeen R -tyypin laadunvalvontakortteihin liitetään soveltuvin osin samat säännöt, kuin $\overline{X}$ -tyypin laadunvalvontakortteihin. Säännöistä enemmän kappaleessa 2.3. [Wheeler92, Shewhart]

Edellä mainittujen tunnuslukujen lisäksi R -tyypin laadunvalvontakortille voidaan laskea erotusten (k kappaletta) keskiarvo $\overline{R}$ (Yhtälö 7) selventämään keskimääräistä poikkeamaa kohdearvosta, eli nollasta.

$$\overline{R} = \frac{\sum_{i=0}^{k} R_i^p}{k}$$

Yhtälö 7 Rinnakkaisten tulosten aritmeettinen keskiarvo

Kuten seuraavasta kuvasta nähdään (Kuva 4), muistuttavat R -tyypin laadunvalvontakortit monilta osin X -tyypin laadunvalvontakortteja.



Kuva 4 Yksinkertainen esimerkki R-kortista [Montgomery]

2.3 Sääntöjoukko

Yksittäinen sääntö on tunnistuskuvio (pattern regocnition) tietylle laadunvalvontakortin pisteiden kuviolle, josta voidaan päätellä, että prosessi ei ole hallinnassa (process is out of control, process is not predictable). Sääntöjoukko on joukko valittuja sääntöjä, joilla prosessia yritetään valvoa valvontakorttien avulla. Alan kirjallisuudessa sääntöjoukoista käytetään muun muuassa termejä: *pattern regocnition, abnormal distribution patterns, sensitizing rules* ja *run rules*. [Montgomery, Wheeler92, Asaka]

Tulosten tulkinta on hyvin prosessikohtaista ja vaatii yleensä huomattavan määrän asiantuntemusta prosessista ja siihen vaikuttavista tekijöistä. Toisaalta, etukäteen valitulla ja tutkitulla sääntöjoukolla voidaan nopeuttaa ja helpottaa varsinaista tulosten tulkintavaihetta. Jos laadunvalvontakorteista tarkkailtaisiin pelkästään yksittäisten pisteiden valvontarajojen ylityksiä tai alituksia, se ei hyödynnettäisi laadunvalvontakorttien ja koko SPC-menetelmän potentiaalia. On esimerkiksi mahdollista, että kaikki pisteet ovat valvontarajojen sisäpuolella, mutta merkittävä osa niistä sijaitsee vain toisella puolelle tavoitearvoa. Tällöin voidaan luonnollisesti epäillä prosessin olevan väärässä tilassa tai asetuksissa. Tätä ei kuitenkaan

havaittaisi, jos tutkittaisiin pelkästään valvontarajojen yksittäisiä ylityksiä. [Montgomery]

Tietotekniikkaa hyväksikäyttäen voidaan tarkistaa nopeasti ja luotettavasti hyvinkin monimutkaisia tunnistuskuvioita. Tarkastuksessa sovellettavan sääntöjoukon valinta ei kuitenkaan ole helppoa, sillä mitä useampia ja monimutkaisempia sääntöjä otetaan mukaan, sitä suuremmaksi kasvaa väärien hälytysten määrä. Suuri määrä vääriä hälytyksiä voi haitata koko prosessia. Sopivan sääntöjoukon valinta on usein myös optimointiongelma.

Eroja X - ja R -valvontakortin sääntöjen tulkinnassa ei ole, mutta luonnollisesti on tärkeä ymmärtää, mitä sääntörikkeet tietyllä korttityypillä tarkoittavat. Yleissääntönä voidaan pitää, että X -kortteja ei tulisi yrittää tulkita, ennen kuin saman muuttujan R -kortti on tulkittu. [Montgomery]

Valmiita sääntöjoukkoja on aikojen saatossa kehitetty useita. Yleensä sääntöjoukko on luotu jonkin tietyn organisaation omaan käyttöön, mutta tietyt, hyväksi havaitut, sääntöjoukot ovat levinneet myös laajempaan käyttöön. Toisaalta, useimmat valmiit sääntöjoukot toimivat hyvin tuotantoprosesseissa, mutta eivät kemiallisissa analyysiprosessissa.

Tunnettuja sääntöjoukkoja ovat esimerkiksi:

- Western Electric, jota kutsutaan myös Zone Rule –nimellä [Montgomery]
- Motorola [Kolarik]
- AT&T [Kolarik]
- Westgard [Westgard]

Näistä ainoa, nimenomaan kemiallisia laboratorioita varten kehitetty sääntöjoukko, on Westgardin sääntöjoukko. Westgardin sääntöjoukko ei itse asiassa ole pelkkä sääntöjoukko, vaan myös tarkasti määritelty prosessi siitä, kuinka sääntöjä tulisi käyttää. Westgardin sääntöjoukossa tutkitaan ensin rikkooko jokin piste $2s$ säännön, eli ylittääkö tulos varoitusrajan (s tarkoittaa keskiarvojen keskiarvon keskihajontaa, katso yhtälö 3). Jos tulos ei ylitä varoitusrajaa, prosessi on hallinnassa. Jos tulos ylittää rajan, tehdään jatkotarkasteluja. [Westgard]

Käytännössä laboratoriot kuitenkin käyttävät usein itse valitsemiaan sääntöjä, jotka sopivat parhaiten heidän omaan prosessiinsa. Sääntöjä voidaan teoreettisesti kehittää miten paljon tahansa, mutta ohessa on listattu joitakin kirjallisuudessa yleisesti mainittuja sääntöjä. Suluissa on mainittu yleensä käytetyt oletusarvot, vaikka ne vaihtelevatkin eri sääntöjoukkojen kesken. [Montgomery, Kolarik, Westgard]

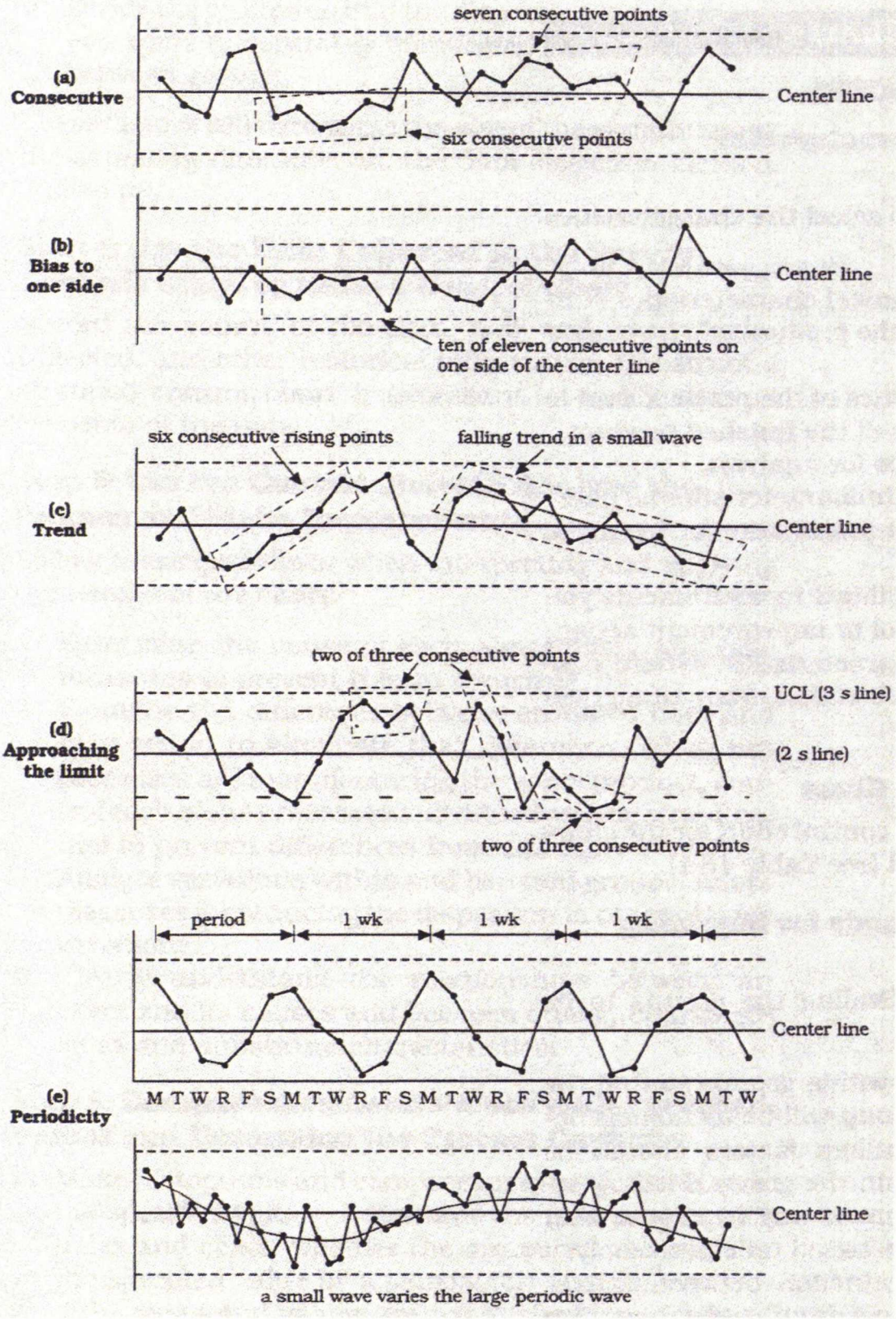
1. N peräkkäistä pistettä $3s$ -rajojen ulkopuolella (N = 1)
2. N peräkkäistä pistettä $2s$ -rajojen ulkopuolella (N = 2)
3. N pistettä K:sta peräkkäisestä pisteestä $2s$ -rajojen ulkopuolella (N = 2, K = 3)
4. N pistettä K:sta peräkkäisestä pisteestä $1s$ -rajojen ulkopuolella (N = 4, K = 5)
5. N peräkkäistä pistettä samalla puolella keskiviivaa (N = 8)
6. N peräkkäistä pistettä nousevassa tai laskevassa trendissä (N = 6)
7. N peräkkäistä pistettä $1s$ -rajojen välissä (N = 15)
8. N peräkkäistä pistettä vaihtaen suunta ylös ja alas vuorotellen (N = 14)
9. N peräkkäistä pistettä jotka eivät ole $1s$ -rajojen välissä (N = 8)

10. Epätavallinen tai ei-satunnainen pistekuvio

11. Yksi tai useampi piste lähellä valvontarajaa

Useimpien sääntöjen seuranta on helppo automatisoida. Käytännössä nykytekniikalla ei kuitenkaan voida täysin korvata asiantuntijoiden tekemiä arvioita laadunvaltakorteista ja prosessin tilasta.

Kuva 5 esittää joitakin yleisimpiä huonosti jakautuneita pistejoukkoja.



Kuva 5 Esimerkkejä huonosti jakautuneista pisteistä [Asaka]

2.4 Esimerkkejä todellisista laadunvalvontatilanteista

Tässä kappaleessa esitetään muutamia asiakkailta kerättyjä todellisia käyttötapauksia, joissa hyödynnetään laadunvalvontakortteja. Suurin osa tiedoista on saatu asiakkaiden kanssa käydyistä keskusteluista projektin kuluessa.

2.4.1 Asiakas A

Asiakas A tutkii päivittäin toimitetuista kymmenistä nestemäisistä näytteistä noin kymmenen ominaisuutta tätä varten hankitulla automaattisella analyysilaitteella. Ominaisuuksia ovat esimerkiksi natriumin ja rasvan määrät näytteessä. Laite tekee yksittäisessä ajossa jokaisesta ominaisuudesta vähintään kaksi rinnakkaista tulosta. Valvontakortit muodostetaan näytteen jokaiselle ominaisuudelle erikseen. Valvontakorttien rajat määritellään aluksi noin 20 tuloksen mukaan, jonka jälkeen korttia aletaan käyttää aktiivisesti laadunvalvonnan apuna. Aika ajoin aktiivinen laadunvalvontakortti suljetaan. Vanhan kortin kanssa rinnakkain kerätään uuteen korttiin taas noin 20 tuloksen joukko, jonka jälkeen vanha kortti suljetaan ja uusi kortti muuttuu aktiiviseksi.

2.4.2 Asiakas B

Asiakas B haluaa noudattaa vain aikaisemmin mainittua Westgardin sääntöjoukkoa laadunvalvontakorttien tulosjoukon tarkastamisessa. Suljettujen laadunvalvontakorttien tulee säilyä kymmeniä vuosia, mahdollisia takuu- ja lakisääteisiä velvoitteita varten.

Asiakas B saa tilauksia omilta asiakkailtaan, jotka haluavat, että lähetetyille näytteille tehdään asiakkaan tilaamat analyysit. Testit voivat olla pitkälle automatisoituja tai käsityönä tehtäviä. Näytteet sisältävät aina rinnakkaisia tuloksia. Tietyissä tapauksissa asiakas B haluaa merkitä analyysin virherajat korttiin. Laadunvalvontanäytteet ovat tutkittavien näytteiden seassa, samassa sarjassa, ja niiden perusteella hylätään tai hyväksytään koko sarja ajon jälkeen. Valvontanäytteet voivat olla laboratorion potilasnäytteistä sekoittamalla valmistettuja referenssimateriaaleja, tai laitevalmistajan toimittamia kontrolliliuoksia. [Kuokkanen]

3 Laadunvalvontakomponentin toteutusprojekti

Tässä kappaleessa kuvataan laadunvalvontakomponentin toteutusprojektin yleiset asiat, kuten vaiheet ja riskit.

3.1 Projektin vaiheistus

Laadunvalvontakomponentin toteutusprojekti, tästä lähtien lyhyemmin QC-projekti, koostui useasta eri vaiheesta, kuten ohjelmistoprojektit yleensä. Nämä vaiheet olivat: [QCpp]

- Projektin aloittamisen hyväksyntä
- Määrittelyvaihe
- Suunnitteluvaihe
- Toteutusvaihe
- Testausvaihe
- Toimitusvaihe

Seuraavissa pääkappaleissa käydään tarkemmin läpi kunkin vaiheen sisältö, joten tässä kappaleessa ei käsitellä vaiheiden sisältöä lukuunottamatta seuraavia huomioita:

- Määrittelyvaihe oli projektin kokonaiskestoon nähden poikkeuksellisen pitkä. Tämä johtui pääosin siitä, että kirjoittajalla ei ollut etukäteistietoa asiakkaiden tavoista käyttää laadunvalvontakortteja. Ainoa tapa hankkia tietoa oli käydä asiakkaiden kanssa keskusteluja, ja käydä henkilökohtaisesti laboratorioissa katsomassa miten laboratoriohenkilökunta laadunvalvontakortteja käyttää. Tähän kului muun tutkimuksen ohella paljon aikaa, joka on laskettu määrittelyvaiheeseen.

- Suunnittelu- ja toteutusvaihe olivat hyvin pitkälle päällekkäisiä. Tämä johtui suunnitteluvaiheen iteratiivisuudesta, jossa yleissuunnittelun jälkeen tietty toiminto suunniteltiin tarkemmin ja toteutettiin välittömästi. Ohjelmistoprosessi ei täysin vastannut perinteistä iteratiivis-inkrementaalia ohjelmistojen kehitysprosessia. Suurin syy tähän oli se, että työtä teki käytännössä yksi ihminen. Siksi ohjelmistoprosessi muistuttaa joiltain osin myös vesiputousmallia.

- Vaikka aktiivinen testausvaihe alkoikin vasta aivan projektin loppupuolella, toteutusvaiheessa pyrittiin tekemään mahdollisimman luotettavaa lähdekoodia ja näin vähentämään toteutusvaiheen testausta. Ideana oli, että koska virheetöntä lähdekoodia ei pystytä projektissa tekemään, virheisiin varaudutaan etukäteen. Luotettavuutta pyrittiin lisäämään muun muassa seuraavilla tavoilla:

    - Lähdekoodiin lisättiin suuri määrä tarkastuksia, joissa pyrittiin tarkastamaan mahdolliset null-arvot, tyhjät kutsuparametrit tai muuten virheelliset arvot. Näitä tarkastuksia tehtiin sekä asiakas-, että palvelinohjelmistoon eri rajapintojen välillä, aiheuttaen näin tuplatarkastuksia. Nämä toimet luonnollisesti lisäsivät koodirivien määrää, mutta luotettavuus nähtiin tärkeämmäksi ominaisuudeksi, kuin lopputuotteen lähdekoodin rivimäärä.

    - Määritettiin hyvin selvät virhekäsittelyt ja virheilmoitukset tietokantaoperaatioiden virhetilanteisiin.

    - Toteutettiin tietokannasta haetun tiedon oikeellisuustarkistukset ja sen perusteella pääteltävissä olevat virheet QC-moduulin käyttämässä tiedossa.

3.2 Riskeistä

Riskejä ei projektin alkuvaiheessa erityisesti kartoitettu. Projektin tutkimus- ja kehitysluonteesta johtuen, projektilla ei katsottu olevan riskejä Software Pointin liiketoiminnan kannalta. Seuraavassa listassa on käyty läpi suhteellisen alkuvaiheessa havainnoituja riskejä ja niiden toteutumista.

- Laboratoriotyöskentelyn ja laadunvalvontakorttien käytön kokemattomuus. Tämä ei kuitenkaan ollut suuri ongelma, koska projektin alkuvaiheessa varattiin runsaasti aikaa asiakaskäyntejä varten

- Java Swing -käyttöliittymäkirjaston käytön kokemattomuus. Projektissa pystyttiin kuitenkin tuottamaan hyvälaatuinen käyttöliittymä, sillä käyttöliittymää on kehuttu selkeäksi sekä Software Pointin työntekijöiden että asiakkaiden taholta.

- Sapphire-järjestelmän puutteellinen tuntemus. Sapphire-järjestelmässä on kattava tietokantamalli ja suuri määrä erilaisia toiminnallisuuksia, jotka tuli ottaa huomioon myös QC-moduulissa. Tekijän puutteellisen Sapphire-järjestelmän tuntemuksen korvasi projektin edetessä Software Pointin työntekijöiden opastus. Tämä riski toteutui hiukan pidempänä toteutusvaiheena.

- Asiakkaiden erilaiset vaatimukset tulivat hyvin selville jo määrittelyvaiheessa. Oli selvää, että kaikkea ei voitu toteuttaa. Periaatteeksi otettiin, että kohtuullisen työmäärän vaativan toiminnon toteuttamiseen vaaditaan ainakin kahden asiakkaan tarve. Riskinä oli se, että tuleeko kompromissinä tehty lopputuote tyydyttämään kaikkia asiakkaita. Suunnitteluvaiheessa pyrittiin ottamaan huomioon eri toiveet tekemällä toiminnoista asiakaskohtaisesti muokattavia. Lopussa voitiin todeta, että asiakkaat olivat tyytyväisiä lopputuotteen toimintoihin ja ilmoituksia toimintojen puuttumisesta tuli yllättävän vähän.

4 Vaatimusten kerääminen

Tässä kappaleessa käydään läpi vaatimusten keräämisprosessi ongelmakohtineen. Tarkemmat QC-projektin vaatimukset löytyvät liitteestä D.

4.1 Vaatimusten keräämisprosessi

Vaikka QC-projekti olikin Sofware Pointin sisäinen projekti, oli luonnollista kerätä vaatimuksia asiakkailta, koska lopputuote tulisi asiakkaiden käyttöön. Lisäksi useimmat Software Pointin asiakkaat ovat asiantuntijoita laadunvalvontakorttien käytössä. Tarkoitus myös oli, että QC-moduuli toimisi asiakkaille toimitetuissa LIMS-järjestelmissä. Lisäksi ajateltiin, että asiakkaat eivät välttämättä haluaisi ottaa käyttöön tuotetta, jonka kehitykseen he eivät ole saaneet tuoda mukaan omia näkemyksiään. Tärkeää oli, että asiakkaat tietävät, että heidät on otettu huomioon laadunvalvontakomponentin suunnittelussa.

Tärkein tietolähde QC-moduulin vaatimuksien keräämisessä oli siis Software Pointin asiakkaat, mutta vaatimuksia pyrittiin keräämään myös muista lähteistä. Erityisesti näihin kuuluivat olemassa olevien laadunvalvontaohjelmistojen läpikäynti, asiakasprojektien projektipäälliköt sekä alan kirjallisuus. [QCurs]

Laadunvalvontakomponentin vaatimusten keräämisprosessi jakautui seuraaviin osiin:

- Projektin perusrajoitusten kerääminen
- Vaatimusten kerääminen kirjallisuudesta (nämä on esitetty tämän työn teoriaosuudessa)
- Vaatimusten kerääminen aiemmissa projekteissa tehdyistä määrittelyistä
- Vaatimusten kerääminen LimsBOSS-tuotteesta. LimsBOSS-tuote on Software Pointin kehittämä LIMS, jossa on QC-moduuli.

- Asiakasvaatimusten kerääminen valitulta joukolta asiakkaita

- Vaatimusten analysointi ja hyväksyntä

Seuraavissa alakappaleissa käydään tarkemmin läpi edellä mainitut keräämisprosessin kohdat.

4.2 Projektin perusrajoitusten kerääminen

Projektin perusrajoitteilla tarkoitetaan projektin alussa tiedettyjä ehdottomia vaatimuksia, jotka näin ollen rajoittivat myös muita vaatimusten keräämisprosessin kohtia ja määrittävät viitekehyksen koko projektille. Projektin perusrajoitteet kerättiin Sapphire-järjestelmän kanssa työskenteleviltä henkilöiltä ja Sapphire-järjestelmän dokumentoitujen rajoitusten avulla.

Projektin perusrajoitteet olivat seuraavat:

- Moduulin tulee toimia sekä www-pohjaisesti että Sapphire LVX – käyttöliittymän kanssa. LVX-käyttöliittymä toimii vain Microsoft Windows -käyttöjärjestelmissä.

- Moduulin tulee toimia sekä Oracle-tietokannan että Microsoft SQL Server -tietokannan kanssa.

- Kielituki (lokalisointituki).

- Moduulin tulee toimia Java sovelmana.

- Moduulin tulee olla Java Swing -käyttöliittymäkirjastolla toteutettu.

- Moduulissa tulee olla tuki Sapphire-järjestelmän käyttöoikeustasoille.

- Tietokantaosaa käyttävät moduulit tulee eriyttää arkkitehtuurisesti muista moduuleista.

Sovelmapohjaisuus tukee Software Pointin valitsemaa linjaa tukea myös asiakkaita, joilla ei ole mahdollisuutta käyttää Sapphire LVX –käyttöliittymää, tuotteen Microsoft Windows -käyttöjärjestelmäriippuvuudesta johtuen. Toisaalta, sovelmana moduuli voi toimia myös Sapphire LVX-käyttöliittymässä, koska LVX-käyttöliittymään voidaan liittää Internet Explorer -selaimessa toimivia komponentteja. QC-moduulin Java-pohjaisuuden perusteena olivat myös arkkitehtuuriset rajoitteet. Sapphire-järjestelmää ajetaan EAServerin päällä, joka pohjautuu Java-kieleen. Sovelmapohjaisuus mahdollistaa myös suhteellisen helpon asennettavuuden asiakaskoneisiin, koska asennus tarvitsee tehdä vain palvelimeen. Asennuksessa on myös ongelmansa, sillä jokaisessa asiakaskoneessa tulee olla Java-virtuaalikone (JRE). Toimitusvaiheessa huomattiin, että joillekin asiakkaille tämä oli ongelma, koska käyttäjillä ei ollut pääkäyttäjäoikeuksia, eivätkä he täten voineet asentaa Java-virtuaalikonetta.

Koska Sapphire-järjestelmä tukee sekä Oracle että Microsoft SQL Server -tietokantoja, tulee myös QC-moduulin tukea niitä. Toteutusvaiheessa tästä aiheutui ongelmia, koska esimerkiksi jotkin SQL-lauseet eivät toimi sellaisinaan molemmissa tietokantajärjestelmässä. Tällöin jouduttiin yleensä tekemään omat SQL-lauseet molemmille tietokantajärjestelmille.

Jotta sovelma olisi käyttöliittymältään tarpeeksi käyttäjäystävällinen ja monipuolinen, sitä ei voitu toteuttaa vanhemmalla AWT-käyttöliittymäkirjastolla. Näin ollen luonnolliseksi vaihtoehdoksi jäi Swing-käyttöliittymäkirjaston käyttö, jolla on mahdollista tehdä hyvinkin näyttäviä ja käyttäjäystävällisiä sovelmia.

Kielituella tarkoitetaan sitä, että kaikki moduulissa käyttäjälle näkyvät tekstit voidaan kääntää käyttäjän kielelle. Yleisemmin ottaen puhutaan lokalisoinnista, jossa kaikki maakohtainen (tai aluekohtainen) informaatio voidaan muuttaa toisen maan vastaaviin versioihin. Näitä ovat esimerkiksi päivämäärien merkitsemistavat ja lukujen muotoilut.

Tietokantaosan eriyttäminen muusta arkkitehtuurista oli tärkeää, koska tulevaisuudessa voi olla mahdollista, että tietyille asiakkaille tehdään räätälöityjä tietokantaosia. Tällöin ei olisi mielekästä, että samalla myös moduulin liiketoimintalogiikka- ja käyttöliittymäosaa tarvitsisi muuttaa.

Käyttöoikeustasoilla rajoitetaan tiettyjen henkilöryhmien oikeuksia tiettyihin QC-moduulin toimintoihin. Näitä käyttöoikeustasoja ovat laadunvalvontakorttien katsominen, muokkaaminen, luominen ja poistaminen. Laboratorioissa useimmat henkilöt ovat oikeutettuja tarkastelemaan laadunvalvontakortteja, mutta yleensä vain tietyillä henkilöillä on oikeus poistaa niitä. Joillakin asiakkailla on esimerkiksi lakiperusteinen velvollisuus osoittaa, että tietyt analyysitulokset täyttävät tietyt laatukriteerit. Laatukriteerien tutkimisessa laadunvalvontakortit ovat yleensä avainasemassa. Näin ollen laadunvalvontakortteja voidaan joutua säilyttämään jopa kymmeniä vuosia ja niiden poistaminen järjestelmästä on tarkkaan valvottua toimintaa.

4.3 Vaatimusten kerääminen aiemmista projekteista

Aikaisempien projektien dokumentteja lukemalla saatiin hiottua monia QC-projektin yksityiskohtia, vaikkakin mullistavia uusia toiminnallisuuksia ei löydetty. Oli mielenkiintoista huomata, että eräässäkin yli kymmenen vuotta vanhassa projektissa oli mietitty paljon samoja asioita, kuin mitä QC-projektissa jouduttiin miettimään. Tärkeimpiä huomioita QC-moduulin kannalta olivat: [Fenix]

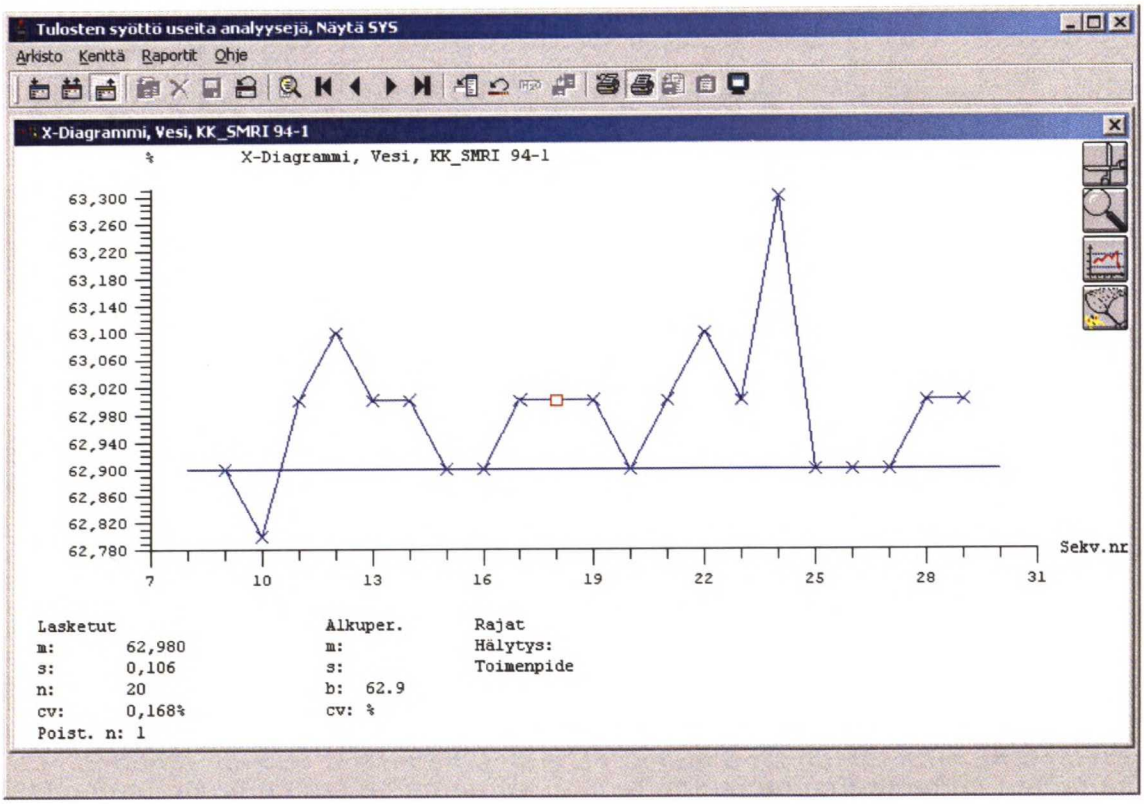
- Laadunvalvontakorteissa tuli olla mahdollista nähdä kaikki poikkeavat tulospisteet eri merkeillä (piste, neliö, kolmio, tms.) ja väreillä. QC-moduulissa hälytys- ja kontrollipisteet ajateltiin näytettäväksi näin jo aikaisemmin, mutta aikaisemman projektin dokumentaatiosta kävi ilmi, että näin kannattaisi tehdä myös tulospisteille.

- Käyttäjän tulisi pystyä valitsemaan laadunvalvontakortin x-akselin asteikko, joko tuloksen sarjanumeron, tai tuloksen syöttöhetken mukaan. Joskus laboratorioissa on tärkeää nähdä tarkka laadunvarmistustuloksen syöttöhetki, mutta toisinaan vain tulosten järjestyksellä on merkitystä.

- X -tyypin laadunvalvontakorteissa tulisi pystyä määrittämään, että halutaanko nähdä vain tulosten erotuksen itseisarvo. Tämä vaatimus itse asiassa toi esille uuden *rr* -tyypin laadunvalvontakortin. Tätä valvontakorttityyppiä ei kuitenkaan aikapulan takia toteutettu QC-moduulin 1.0.0 versioon.

4.4 Vaatimusten kerääminen LimsBOSS-järjestelmästä

LimsBOSS-järjestelmä on Software Pointin kehittämä ja toimittama LIMS-järjestelmä. Oheisessa kuvassa (Kuva 6) on esimerkki LimsBOSS-järjestelmän X -tyypin laadunvalvontakortista. LimsBOSS-järjestelmä pystyy laadunvalvontakorttien osalta lähes kaikkeen siihen, mitä useimmat asiakkaat tarvitsevat.



Kuva 6 X -tyypin laadunvalvontakortti LimsBOSS-järjestelmässä

LimsBOSS-järjestelmä on ollut jo vuosien ajan tuotantokäytössä sadoilla eri asiakkailla. Järjestelmästä kerättiin toiminnallisuuksia ja kehitysehdotuksia suoraan

sitä tutkimalla sekä asiakkaiden vuosien aikana antaman palautteen avulla. Näistä havainnoista QC-projektin kannalta oleellisimmat olivat:

- Tuloksia tulisi voidaan poistaa tilastollisista laskennoista.
- Käyttäjä voi valita x-akselille, joko N viimeistä tulosta, tai jonkin tietyn aikavälin.
- Tarkennus (zoom) käyttäjän valitsemalla alueelle laadunvalvontakortissa.
- Leikkaa-liitä –toiminnallisuus (copy-paste). Valitaan jokin tietty alue laadunvalvontakortista, joka halutaan kopioida leikepöydälle. Myöhemmin tämä alue voidaan esimerkiksi liittää kuvana toiseen dokumenttiin.
- Laadunvalvontakorttien pisteitä tulee pystyä tutkimaan niiden matemaattisesti tarkkojen arvojen kanssa. Tämä on mahdollista esimerkiksi tooltip-ominaisuuden avulla, jossa hiiren kohdistimen alla olevalle pisteelle luodaan lisätietokenttä pisteen tarkoista matemaattisista arvoista.
- Käyttäjän tulee voida nähdä erillinen tilastollinen näkymä, jossa on kaikki yleisimmin tarvittavat tilastolliset tunnusluvut laadunvalvontakortin tuloksista. Nämä tunnusluvut tulisi laskea kaikille laadunvalvontakortin tuloksille ja mahdollisesti valituille osajoukolle.
- R-korttien laskennassa tulisi ottaa huomioon rinnakkaisten tulosten määrä neliöjuuren jakajassa. Tämä on merkittävä asia, sillä mitä useampaa rinnakkaista tulosta käytetään, sitä pienemmäksi virherajat käyvät ja sitä lähempänä tulosten tulisi olla kohdearvoa.

4.5 Asiakasvaatimusten kerääminen asiakkailta

Tämän projektin puitteissa asiakasvaatimuksilla tarkoitetaan asiakkaiden esittämiä toivomuksia tuotteesta. Projekti oli kuitenkin Software Pointin sisäinen projekti,

jolloin projektin sisällöstä voitiin päättää itsenäisesti. Käytännössä asiakkaiden toiveilla oli kuitenkin hyvin suuri painoarvo vaatimuksia kerättäessä.

Asiakasvaatimusten kerääminen tehtiin valitulta joukolta WSP:n asiakkaita. Asiakkaat valittiin seuraavin perustein:

- Valituilla asiakkailla tuli olla käytössä toimiva laadunvalvontajärjestelmä. Tämä laadunvalvontajärjestelmä saattoi käytännössä tosin olla varsin yksinkertainen, esimerkiksi mappirivistö millimetripaperille piirrettyjä laadunvalvontakortteja ja laadunvalvontamateriaalin analysointiin sovitettu manuaalinen prosessi. Toimiva, vaikkakin yksinkertainen, laadunvalvontajärjestelmä takaisi kuitenkin sen, että asiakkaalta löytyisi käytännön asiantuntemusta laadunvalvontajärjestelmän toiminnasta.

- Asiakkaalla tuli olla kiinnostusta ottaa käyttöön toteutettava laadunvalvontakomponentti.

- Asiakkaalla tuli jo olla käytössä, tai olla lähiaikoina tulossa käyttöön, Sapphire-järjestelmä. Tämä takaisi sen, että toteutettava laadunvalvontakomponentti toimisi asiakkaan LIMS-järjestelmässä.

Näillä perusteilla asiakkaiksi valittiin:

- Rautaruukki
- Valio
- Yhtyneet Laboratoriot
- KTL (Kansanterveyslaitos)
- GTK (Geologian Tutkimuskeskus)

- Tullilaboratorio
- Central Soya

Kokonaisuutena ajatellen asiakasvaatimusten kerääminen oli vaativa tehtävä, mutta se antoi toteutukselle luotettavuutta. Vaativaksi keräämisen teki asiakkaiden suuri määrä suhteessa projektin resursseihin ja eri asiakkaiden toisistaan poikkeavat LIMS-konfiguroinnit. Samoista asioista puhuttiin myös eri termein eri laboratorioissa.

4.6 Vaatimusten analysointi ja hyväksyntä

Kerättyjä vaatimuksia analysoitiin keräyksen edetessä, kuten myös erillisissä kokouksissa, joissa käsiteltiin vaikeimpia esille tulleita kysymyksiä. Projektia varten perustettiin alussa seurantaryhmä, johon kuului noin kymmenen Software Pointin työntekijää. Ryhmän jäsenillä oli yhdessä paljon käytännön tuntemusta laboratoriotyöstä, käytetystä Sapphire-järjestelmästä ja asiakkaiden tarpeista. Lisäksi ryhmään kuului teknisiä asiantuntijoita ja yrityksen johtotason henkilöitä. Tämän seurantaryhmän yhteisissä keskusteluissa, joihin kutsuttiin tarpeen mukaan myös muita, selvisi moni vaikea asia.

Tällainen oli esimerkiksi tilanne, jossa pohdittiin pitääkö sallia laadunvalvontadiagrammin hälytys- ja kontrollirajojen muutos aina kun käyttäjä niin haluaa. Tämän toiveen oli esittänyt eräs asiakas, mutta se olisi ollut erittäin vaikea toteuttaa Sapphire-järjestelmän puitteissa. Niinpä asia otettiin käsiteltäväksi eräässä seurantaryhmän tapaamisessa, jossa sitten päätettiin ottaa käyttöön laadunvalvontakortin tila-attribuutti. Tiloja määriteltiin kolme: keräys, aktiivinen ja suljettu. Keräystilassa käyttäjä voi muuttaa rajoja jatkuvasti, mutta aktiivisessa tilassa ei. Tämä tila-ajattelu vastaa myös hyvin laadunvalvontakorttien teoreettista mallia.

Muiden laadunvalvontajärjestelmien sääntöjoukkojen toimintaa analysoimalla huomattiin, että ne olivat usein liian kiinteästi määrättyjä, eikä käyttäjä päässyt vaikuttamaan niihin. QC-moduuliin päätettiin tehdä hyvin muokattava sääntöjoukko,

jossa minkä tahansa säännön voisi tiputtaa pois ja sen vakioita määrittää vapaasti. Näin mahdollisimman moni asiakas saisi vapaasti muokata sääntöjoukon toimintaa omaan laadunvalvonprosessiinsa sopivaksi.

Vaatimusmäärittelyä lähetettiin tietyille asiakkaille useita kertoja kommentoivaksi. Näiden kommentointien jälkeen siihen tehtiin yleensä myös joitain muutoksia. Vaatimusmäärittelyn katselmointi ja hyväksyttäminen seurantaryhmällä oli viimeinen vaihe vaatimusten keräämisprosessissa.

5 Suunnitteluvaihe

Suunnitteluvaiheessa määriteltiin muun muassa QC-moduulin arkkitehtuuri, tietokantarakenne, ja kehitettiin käyttöliittymää kohti lopullista muotoaan. Suunnitteluvaihe ei kuitenkaan ollut erillinen vaihe projektissa, vaan se limittyi hyvin paljon toteutusvaiheen töiden kanssa, kuten kappaleessa 3 on kuvattu. Suunnitteluvaihe oli iteratiivinen: ensin valittiin pienehkö toiminnallisuus, joka suunniteltiin ja sitten toteutettiin. [QCsds]

Suunnitteluvaiheen merkittäviä tapahtumia olivat:

- Diagrammien piirtokomponentin etsintä
- Liian suurten jar-tiedostojen ongelma
- Prototyyppi
- Arkkitehtuuri
- Käyttöliittymäsuunnittelu

Seuraavissa alakappaleissa kuvataan edellä mainittuja tapahtumia.

5.1 Diagrammien piirtokomponentin etsintä

Jo aivan QC-pjojektin alussa oli selvää, että itse laadunvalvontadiagrammien peruspiirtotoimintoja ei kannata toteuttaa alusta asti projektin omilla resursseilla. Markkinoilla on suuri määrä valmiiksi tehtyjä yleistason piirtokomponentteja, joita QC-moduulin toteutuksessa voitaisiin käyttää. Ongelmaksi jäi sopivan piirtokomponentin etsiminen.

Diagrammin piirtokomponentin etsintä aloitettiin jo määrittelyvaiheessa, mutta varsinainen päätös piirtokomponentin valinnasta tehtiin vasta suunnitteluvaiheessa,

kun piirtokomponentin etsinnästä vastannut henkilö sai tutkimuksensa valmiiksi. [Liite A ja B]

Tutkimuksessa tuli ottaa huomioon muun muassa seuraavat piirtokomponentin ominaisuudet:

- Tukee tulostusta
- Tukee diagrammien näkyvien osin värien ja piirtotyylien muuttamista
- On toteutettu Swing-käyttöliittymäkirjaston avulla
- Toimii Java 1.4.1 –version kanssa
- Tukee tarkennusta (zooming)
- Joustava omille muutoksille
- Lähdekoodi saatavissa omaan käyttöön

Tutkimuksessa löydettyjen vaihtoehtoisten piirtokomponenttien arvioinnin ja vaatimusta perusteella, sopivaksi piirtokomponentiksi valittiin jFreeChart-komponentti, joka on julkaistu Lesser GNU Public License -lisenssin (LGPL-lisenssi) alla.

Valittu komponentti tuki jo itsessään useimpia tärkeimmistä vaatimuksista. Lyhyellä katselmoinnilla todettiin lähdekoodi ja dokumentointi laadukkaaksi. Lisäksi komponentin arkkitehtuurissa käyttöliittymän esitysmuoto oli eriytetty tietomallista, kuten Swing-käyttöliittymäkirjaston arkkitehtuurissa (MVC-malli). Nämä ominaisuudet tekivät piirtokomponentin lähdekoodin hyvin muutoksia sietäväksi. Komponentin puolesta puhui myös aktiivinen kehitystyö ja keskusteluryhmät, joista sai helposti apua ongelmatilanteissa. Keskusteluryhmien asiantuntemusta käytettiinkin ahkerasti QC-projektin aikana.

5.2 Liian suurten jar-tiedostojen ongelma

Java-sovelmien toiminta perustuu siihen, että palvelimelta ladataan verkon yli asiakaskoneelle tarvittavat komponentit jar-paketteina, joita sitten suoritetaan asiakaskoneen selaimella Java-virtuaalikoneessa. Näitä verkon yli ladattavia komponenttejä QC-moduulissa olivat itse QC-moduulin komponentti, jFreeChartin kaksi piirtokomponenttia ja Sapphire-järjestelmän mukana tuleva komponentti. Sapphire-järjestelmän komponentillä saadaan yhteys tietokantaan, ja sen avulla voidaan käyttää muita Sapphire-järjestelmän tarjoamia palveluita.

Näistä komponenteistä Sapphire-järjestelmän komponentti oli koolta yli kymmenen megatavua. Näin ollen sen lataaminen verkon yli asiakaskoneelle olisi kestänyt liian kauan, jotta käyttäjä olisi jaksanut odottaa sovelman käynnistymistä.

Tämän ongelman ratkaisemiseksi tehtiin tutkimus mahdollisista keinoista ladattavien jar-tiedostojen koon pienentämiseksi. Suurin osa Sapphire-komponentin luokista oli turhia QC-moduulin kannalta. Asiaa tutkinut henkilö löysikin helppokäyttöisen keinon jar-tiedostojen pienentämiseen. Menetelmässä komponentistä poistetaan kaikki se mitä QC-moduuli ei käytä. Tähän löytyi internetistä sopiva ilmainen työkalu nimeltään ProGuard, joka osaa annettujen alkutietojen perusteella poistaa käyttämättömät Java-luokat komponentistä. Tällä keinolla jar-tiedoston kooksi saatiin noin puoli megatavua, joka on käytettävyyden kannalta hyväksyttävä koko.

Jatkosuunnitelmissa on päästä kokonaan eroon Sapphire-järjestelmän komponentistä. Alustavana suunnitelmana on ollut asentaa Sapphire-järjestelmän komponentti ainoastaan palvelimelle. Lisäksi tarvittaisiin erillinen muunnoskomponentti, joka toimisi asiakaskoneen ja Sapphire-järjestelmän komponentin välissä. Tällöin asiakaskoneelta kutsuttaisiin muunnoskomponentin tarjoamia palveluita, joka kutsuisi Sapphire-järjestelmän komponentin palveluita ja välittäisi sen antamat tulokset takaisin asiakaskoneelle. Aihe vaatii kuitenkin lisätutkimuksia.

5.3 Prototyyppi

Suhteellisen aikaisessa kohdin määrittelyvaihetta selvisi, että olisi järkevää tehdä QC-moduulista prototyyppi. Tähän oli kaksi merkittävää syytä:

- Sapphire-järjestelmän toimintojen käytöstä sovelman kautta oli hyvin vähän kokemusta.

- Swing-grafiikkakirjaston käytöstä ei ollut merkittävää kokemusta, ja haluttiin nähdä pystytäänkö sillä toteuttamana halutut toiminnallisuudet.

Prototyypin tuli osoittaa Sapphire-toimintojen toimivuus, vahvistaa arkkitehtuurisuunnittelun perusteet, ja toimia lähtökohtana lopun järjestelmän toteuttamiselle.

Jälkeenpäin huomattiin prototyypin toteutuksella olleen ainakin seuraavat hyödyt:

- Saatiin selvennettyä luokkarakennetta. Suunnitteluvaiheen ongelmana oli luokkasuunnittelun vaikeus. QC-projektia ei lähdetty alusta asti toteuttamaan olioanalyysiperiaatteiden (OOA) mukaisesti, vaikkakin joitakin oliosuunnittelun (OOD) periaatteita hyödynnettiin. Koska analyysivaihetta ei tehty käyttötapausten (use cases) pohjalta, ei myöskään luokkasuunnittelua voitu käyttötapauksista johtaa. Jäljelle jäi joko subjektiivisesti tutkia vaatimusmäärittelyä, ja sitä kautta päätellä luokkarakenteet, tai tutkia vaatimusmäärittelyä jonkin muun tunnetun menetelmän avulla. Alkuvaiheen suunnittelun jälkeen kuitenkin selvisi, että ei ollut mahdollista varmistua siitä, että riittäisikö mikään tutkituista menetelmistä luomaan tarpeeksi tarkkaa käsitystä tarvittavasta luokkarakenteesta. Näin ollen päädyttiin prototyypin toteuttamiseen. . Prototyypin kehitysvaiheessa useimmat luokkarakenteet ikään kuin löysivät paikkansa. Osittain tämä johtuu varmaan myös siitä, että käytettävät ohjelmointitekniikat ja komponentit tulivat tutuiksi.

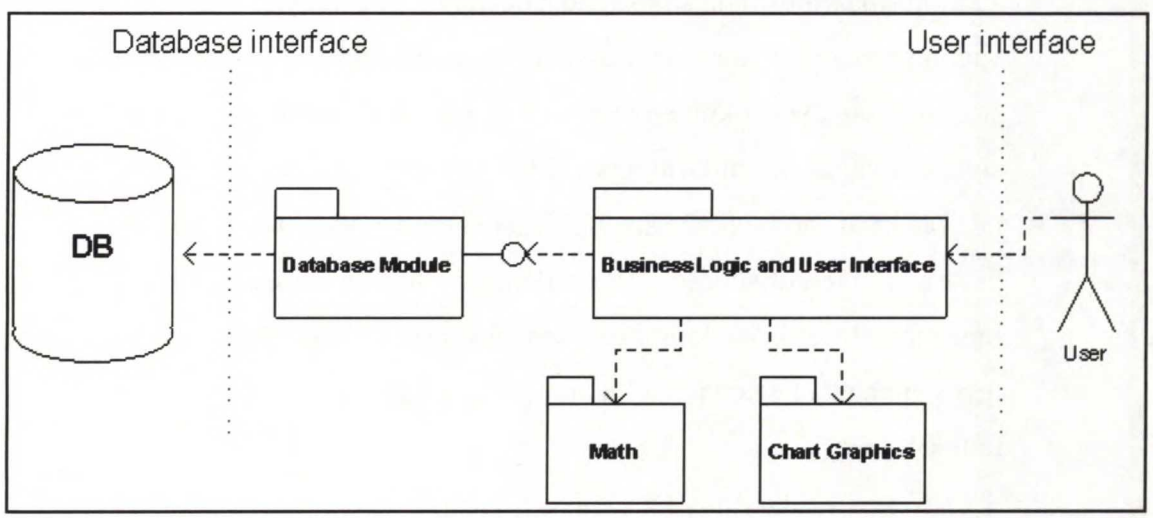
- Saatiin varmuus Sapphire-järjestelmän tarjoamien tietokantaoperaatioiden toimivuudesta QC-sovelmassa.

- Saatiin lisää todisteita arkkitehtuuriratkaisujen oikeellisuudesta, kuten tietokantaosan eriyttämisen tarpeesta. Tietokantaosan eriyttämistä tuki jo pelkästään tieto siitä, että QC-moduulin versio 1.0.0 joutuisi tukemaan kahta tietokantajärjestelmää.

- Saatiin rakennettua toimiva ketju tietokannasta liiketoimintalogiikan kautta aina yksittäisen laadunvalvontakortin käyttöliittymäosaan. Näin saatiin toimiva pohja, jonka päälle oli hyvä lähteä rakentamaan lisää toiminnallisuuksia.

5.4 Arkkitehtuurisuunnittelu

Arkkitehtuuriksi ajateltiin aivan QC-projektin alkuvaiheessa kolmitasoarkkitehtuuria, jossa siis erotetaan toisistaan tietokanta-, liiketoimintalogiikka- ja käyttöliittymäosat (Kuva 7). Tietokantaosan erottaminen tiedettiin pakolliseksi jo ennen projektin alkua, sillä osa ominaisuuksista saattaa tulevaisuudessa tulla asiakaskohtaisiksi, jolloin tietokantaosan muokkaaminen on todennäköistä. Tällöin tietokantaosan tulee olla helposti vaihdettava ilman, että se aiheuttaa muutoksia muissa arkkitehtuurin osissa.



Kuva 7 Yleistason arkkitehtuuri

Liiketoiminta- ja käyttöliittymäosan erottaminen huomattiin kuitenkin vaatimusmäärittelyvaiheessa turhaksi. Tulevaisuudessa ei todennäköisesti tule tilannetta, jossa osien erottaminen olisi hyödyllistä. Tarvittavan työmäärän suhde hyötyyn oli liian suuri.

5.5 Käyttöliittymäsuunnittelu

Käyttöliittymän toimivuus otettiin aivan projektin alussa erääksi tärkeimmäksi vaatimukseksi. Tämä siksi, että osa laboratorioiden henkilöstöstä saattaa olla melko kokemattomia tietokoneiden käytössä, ja näin ollen käyttöliittymän tulee olla helppo ja kätevä käyttää. Jo projektin vaatimusmäärittelyosassa tehtiin ensimmäinen paperiversio käyttöliittymästä. Tästä paperiprototyypistä oli hyötyä koko suunnitteluvaiheen ajan. Suunnitteluvaihe oli suuresti käyttöliittymäsuunnittelun ohjaama. Tämä oli toisaalta luonnollista, sillä suurin osa QC-moduulin toiminnoista liittyy käyttöliittymään ja grafiikkaan.

Käyttöliittymäsuunnittelussa ongelmana oli se, että yleensä sovelmat ovat pieniä, yhden ikkunan sovelluksia. Oli hankalaa löytää esimerkkejä laajoista sovelmista, jotka käyttäytyisivät kuten perinteiset ikkunapohjaiset ohjelmat. Lopulta päädyttiin näyttö-pohjaiseen (screens) ikkunointimalliin, jossa uusi näyttö avautui aina edellisen päälle sen mukaan, mitä käyttäjä teki. Tällä saatiin käyttöliittymä muistuttamaan Sapphire-järjestelmän selainpohjaista käyttöliittymää, jossa ei Windows-tyyppisesti aukea aina uutta ikkunaa jokaiselle toiminnolle.

6 Toteutus

Tässä kappaleessa käydään läpi toteutusvaiheen tapahtumia ongelmineen.

6.1 Toteutusvaiheen rakenne

QC-moduulia lähdettiin toteuttamaan pienissä osissa iteratiivisesti yhdessä suunnitteluvaiheen kanssa. Toteutusvaiheen ensimmäiset viikot menivät sovelman grafiikkakontrollin luomiseen.

Seuraavaksi tehtiin pääsivusto, johon oli helppo lisätä muita toiminnallisuuksia. Ensimmäisenä näistä toteutettiin $\overline{X}$ -tyypin laadunvalvontakortti. Jotta sivuille saataisiin mielekästä näytettävää, täytyi samalla toteuttaa myös tietokantaosaa. Koko projektia ajatellen tietokantaosuuden toteutus vei suurimman osan ajasta. Tämä johtui muun muassa siitä, että aina ei voinut suoraan käyttää SQL-lauseita, vaan välillä oli pakko käyttää Sapphire-järjestelmän tarjoamia tietokantapalveluja. Lisätyötä aiheutti myös projektin aikana useaan otteeseen muuttunut tietokantarakenne.

Seuraavaksi QC-moduulia laajennettiin ymmärtämään perustasolla myös R -tyypin laadunvalvontakortteja. Samalla $\overline{X}$ -tyypin laadunvalvontakortteja laajennettiin siten, että niiden toiminnallisuus alkoi jo muistuttaa lopullista versiota, esimerkiksi sääntöjen ja hälytysmerkkien osalta. Tämän jälkeen toteutettiin erikoisempia toiminnallisuuksia kuten tulostus, erilaisia informaatiopaneeleita korttien ominaisuuksista, tilakontrolli ja vaativia algoritmejä. Haastavimpia olivat erilaisten hälytyskuvioiden tunnistamiseen tarkoitetut algoritmit ja perusmatematiikan (esimerkiksi keskihajonta ja raja-arvolaskenta) toteuttaminen mahdollisimman virheettömästi, kaikki erikoistapaukset huomioon ottaen.

Loppuvaiheessa päätettiin aikatauluongelmien vuoksi jättää *rr* -tyypin laadunvalvontakorttien tuki pois. Loppuaika käytettiin jo toteutettujen toiminnallisuuksien viimeistelyyn.

Dokumentointia tehtiin samalla toteutuksen kanssa, jolloin dokumentit pysyivät hyvin ajan tasalla. Päivitettyjä dokumenttejä olivat:

- Projektisuunnitelma
- Vaatimusmäärittely
- Suunnitteludokumentti
- Käyttäjämanuaali (sisältää asennusohjeet)

Erityisesti käyttäjämanuaalin päivittäminen toteutusvaiheen aikana oli hyvä idea, koska ominaisuudet ja taustalla olevat tekniset ratkaisut olivat tuoreena mielessä.

6.2 Toteutusvaiheen ongelmia ja loppuarviointi

Toteutusvaiheessa tuli vastaan useita ongelmia, joista osaa oli lähes mahdotonta ennustaa etukäteen. Seuraavassa tarkastellaan joitakin ongelmista.

6.2.1 Diagrammien hakemistorakenne

Suhteellisen alkuvaiheessa toteutusvaihetta selvisi, että laadunvalvontakorttien ryhmittelyä ei oltu ajateltu tarpeeksi pitkälle. Käyttäjän tulee voida ryhmitellä laadunvalvontakortteja haluamallaan tavalla. Tällaisia ryhmittelyjä voisivat olla esimerkiksi laboratorioiden mukaiset jaot, vanhat ja uudet laadunvalvontakortit tai laboratoriolaitteiden mukainen ryhmittely. Asiakaskeskusteluissa selvisi, että ei olisi mahdollista löytää kaikille asiakkaille sopivaa kiinteää ryhmitysrakennetta, joten täytyi löytää helposti muunneltava ratkaisu eri asiakkaiden tarpeisiin.

Suunnittelupalaverin jälkeen vaihtoehdoiksi löytyivät seuraavat ratkaisut:

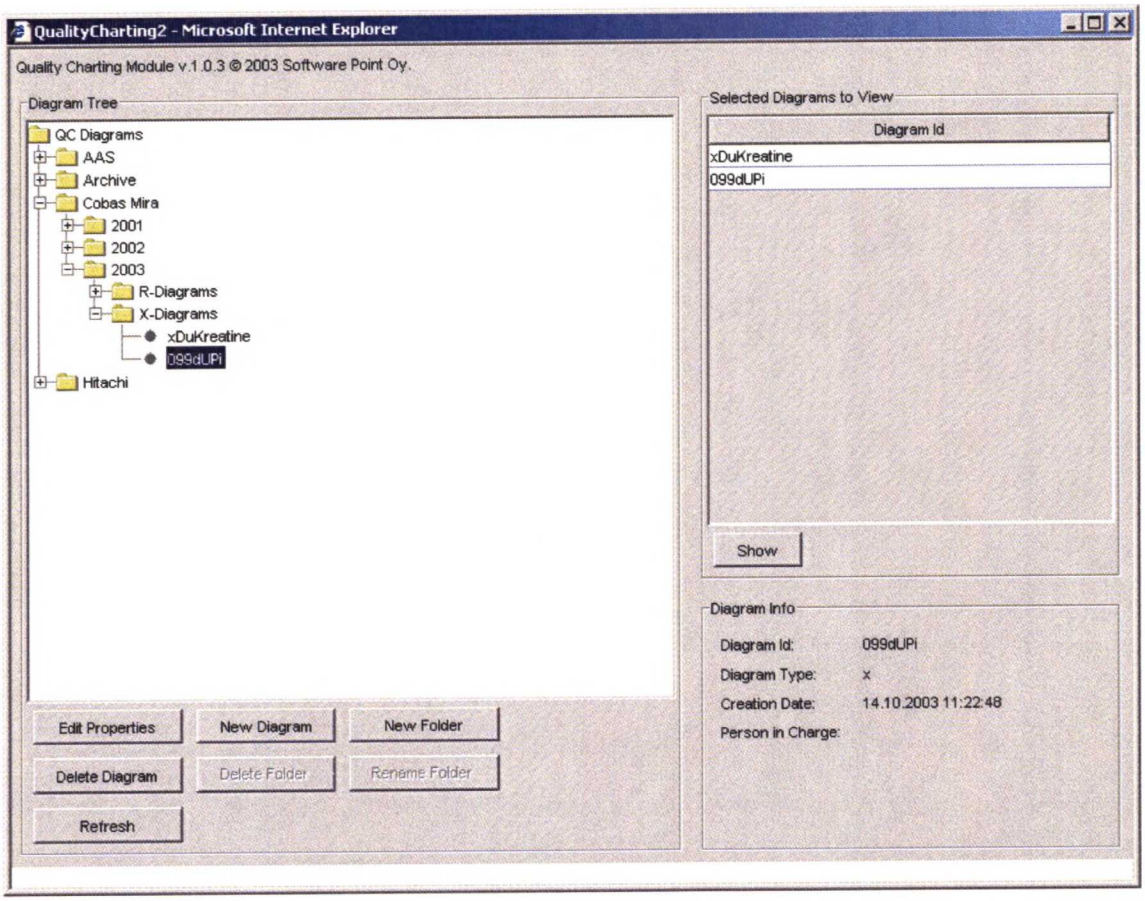
- Sapphiren kategorioiden mukainen ryhmittely
- Swing-käyttöliittymäkirjaston JTree-komponentin suora serialisointi kantaan

- Hakemistorakenne uudella tietokantataululla

Sapphiren kategoriat toimivat pitkälti samanlaisesti kuin yleisesti tunnettu hakemistorakenne, mutta sitä kontrolloi Sapphiren oma tietokantarakenne. Sapphiren kategorioiden käytössä oli kuitenkin paha puute, sillä se sallii vain yksitasoisen hakemistorakenteen. Tämän ei katsottu riittävän QC-moduulin tarpeisiin. Lisäksi kategorioiden hallinta olisi tapahtunut Windows-pohjaisella LVX-työkalulla.

Swingin JTree –komponentin serialisointi tietokantaan olisi ollut helppo toteuttaa, mutta kyseessä ei olisi ollut mielekäs tapa käyttää tietokantaa. Tällä menetelmällä olisi menetetty kaikki Sapphire-järjestelmän tarjoamat palvelut, kuten käyttöoikeustasot.

Ainoaksi hyväksi vaihtoehdoksi jäi tiedostojärjestelmistä tuttu hakemistorakenne. Siinä käyttäjä voi luoda puumaisesti hakemistoja, joihin hän voi sijoittaa halutut laadunvalvontakortit. Jokainen laadunvarmistuskortti voi kuulua vain yhteen hakemistoon. Alla oleva kuva (Kuva 8) näyttää kuvitteellisen hakemistorakenteen. Siinä laadunvalvontakortit on ensin jaettu laitetasolla, sitten vuositasolla ja lopulta korttityypeittäin.



Kuva 8 Esimerkki QC-moduulin hakemistorakenteesta

Tällaisen hakemistorakenteen ylläpitäminen vaati uuden taulun kantarakenteeseen. Onneksi uusi taulu oli irrallaan muusta hakemistorakenteesta, joten muutoksia jo toteutettuihin osiin ei tarvinnut tehdä.

Tämän jälkeen ongelmaksi jäi se, että mitä operaatioita hakemistorakenteen hallintaan toteutetaan. Osa operaatioista on melko monimutkaisia ja niiden toteuttaminen olisi saattanut viedä kohtuuttoman paljon aikaa suhteessa niistä saatavaan hyötyyn. Mahdollisia tarvittavia operaatioita ongelmakohtineen olivat muun muassa:

- Uuden hakemiston luonti: voidaanko hakemisto luoda minkä tahansa hakemiston alle ja miten toteutetaan hakemistojen käyttöoikeustasojen hallinta.

- Hakemiston tuhoaminen: rekursiivinen vai ei, tuhoaako se myös laadunvalvontakortit ja miten toteutetaan hakemistojen käyttöoikeustasojen hallinta.

- Hakemiston siirtäminen toiseen hakemistoon: onnistuuko laadunvalvontakortteja sisältävän hakemiston liikuttaminen ja hakemistojen käyttöoikeustasojen hallinta.

- Hakemiston uudelleen nimeäminen: miten toteutetaan hakemistojen käyttöoikeustasojen hallinta.

Versioon 1.0.0 toteutettiin uusien hakemistojen luonti, hakemiston uudelleen nimeäminen ja hakemiston tuhoaminen. Hakemiston tuhoamisessa tulee hakemiston olla tyhjä muista hakemistoista ja laadunvalvontakorteista. Tällä vältetään vahingossa tapahtuvat rekursiiviset tuhoamiset, sillä laadunvalvontakorttien tuhoaminen on kuitenkin harvinainen toimenpide. Käyttöoikeustasojen hallinta hoidettiin siten, että käyttäjä voi muuttaa kaikkea hänelle näkyvää niillä käyttöoikeustasoilla, jotka hänelle on annettu.

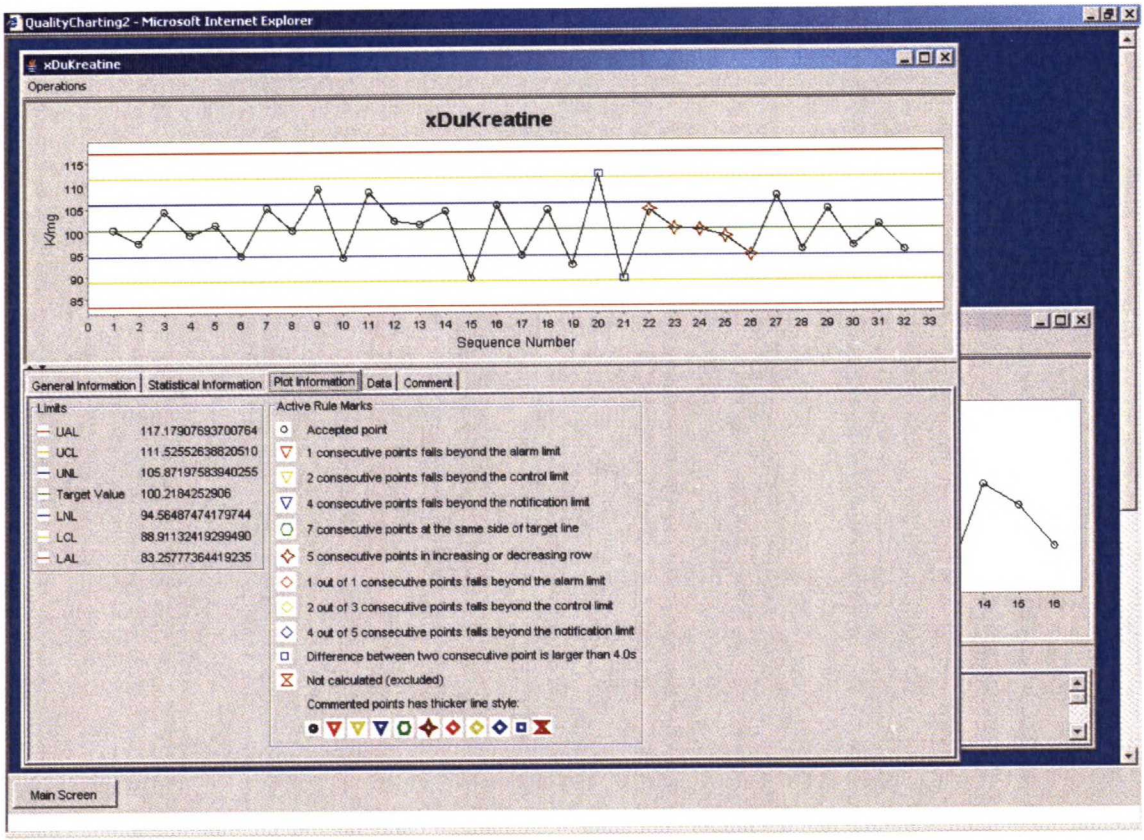
6.2.2 jFreeChartin ongelmia

Hankitun kaupallisen grafiikkakirjaston, jFreeChartin, mukana tuli paljon hyödyllisiä ominaisuuksia, mutta myös ongelmia.

Hieman liiankin monipuolinen jFreeChart-komponentti lisäsi verkon yli ladattavien komponenttien kokonaismäärää yli 500 kilotavulla. Tosin tätä latauskokoa pystyttiin vähentämään ProGuardin avulla samalla tavoin kuin Sapphire-järjestelmän jar-paketin tapauksessa (katso kappale 5.2).

Toisaalta JfreeChartilla saatiin aikaan erinomaista grafiikkaa, kuten alla olevasta esimerkkikuvasta (Kuva 9) nähdään. Kyseisessä kuvassa on valvontakorttien piirtoalueella kaksi korttia päällekkäin. Päällimmäisenä olevasta *xDuKreatine*-nimisestä kortista nähdään, että siihen on piirretty kaikki kuusi valvontarajaa.

Lisäksi nähdään, että laadunvalvontatulokset rikkovat tulosvälillä 20 - 26 kahta eri sääntöä.



Kuva 9 Kaksi laadunvalvontakorttia oheistietoineen QC-moduulin piirtämänä

6.2.3 Muita ongelmia

Jo suunnitteluvaiheessa huomattiin, että Sapphiren WWW-liittymän vaatimien pakollisten reunavalikoiden jälkeen itse QC-sovelmalle ei jäänyt paljon tilaa.

Tämä olisi ongelma, jos sovelmaa käytettäisiin samassa ikkunassa kuin mistä se Sapphire-järjestelmän WWW-liittymässä avattiin. Ongelmaa pahensi se, että QC-moduulin tarkoituksena on näyttää laadunvalvontakortteja, jotka tarvitsevat erityisen runsaasti tilaa ollakseen hyödyllisiä.

Ratkaisuksi löytyi oman ikkunan avaaminen. Näin ollen käyttäjä voisi itse päättää QC-moduulin viemän tilan muuttamalla ikkunan kokoa. Selaimesta avattu uusi ikkuna on niin sanottu ponnahdusikkuna (pop-up –ikkuna), jossa on tiettyjä

ongelmia. Ponnadusikkunoiden avaaminen voi esimerkiksi olla kiellettyjä joissakin ympäristöissä. Tähän mennessä asiakkailta ei kuitenkaan ole vielä kuulunut vastustusta ponnahdusikkunoiden käyttöön.

Tilan lisäämiseksi myös itse QC-sovelmassa pyrittiin maksimoimaan laadunvalvontakorttien näyttötila. Tilan lisäämiseksi painikerivejä vähennettiin ja toimintoja siirrettiin alasvetovalikkoon.

Toinen suurempi ongelma oli käyttöliittymän ohjelmointi käsin, joka oli hidasta, koska käyttöliittymästä tuli suhteellisen monimutkainen. Projektissa ei tarkoituksella käytetty automaattisesti lähdekoodia generoivia käyttöliittymätyökaluja kuten JBuilderiä. Käyttöliittymän tekeminen käsityönä mahdollisti grafiikkakirjaston kaikkien ominaisuuksien tehokkaan hyödyntämisen. Lisäksi automaattisesti ohjelmakoodia generoivat käyttöliittymätyökalut yleensä joko jättävät osan toiminnallisuuksista tukematta tai generoivat koodia virheellisesti. Esimerkiksi JBuilder ei tuottanut testeissämme tarpeeksi laadukasta koodia.

6.2.4 Laadunvarmistustulosten erikoistapauksia

Suunnitteluvaiheessa ei otettu huomioon tilannetta, jossa yksittäiselle laadunvarmistustulokselle annetaan poikkeavia ominaisuuksia. Näitä ominaisuuksia ovat:

- Onko tulos diagrammeissa näkyvä (visible)
- Otetaanko tulos mukaan laskentoihin (calculated)
- Tuloksen vapaamuotoinen tekstikenttä (comment)

Nämä ovat kaikki tietoja, joita Sapphire-järjestelmässä ei voida yksittäiselle laadunvarmistustulokselle antaa. Asian tutkimisen jälkeen selvisi, että vaihtoehtoisia ratkaisutapoja on kaksi:

- Luodaan uusia sarakkeita Sapphire-tietokantamallin tulostauluun, jossa kaikki laadunvarmistutulokset ovat.

- Lisätään uusi taulu QC-moduulin tietokantarakenteeseen, joka sisältäisi ne laadunvarmistustulokset, joissa on poikkeavia ominaisuuksia.

Uusien sarakkeiden luominen Sapphire-järjestelmän tulostauluun ei olisi ratkaissut asiaa kokonaan, koska tällöin laadunvarmistustuloskohtaisista tiedoista ei olisi tullut laadunvalvontakorttikohtaisia. Esimerkiksi jos käyttäjä laittaa diagrammin D1 tulokselle numero neljätoista vapaamuotoisen tekstikommentin, niin tämä sama kommenttiin ei tulisi näkyä diagrammissa D2, joka myös käyttää tulosta numero neljätoista. Sapphire-järjestelmän tulostaulussa samoja tuloksia ei voida jaotella laadunvalvontakorttikohtaisesti. Näin ollen ainoaksi vaihtoehdoksi jäi oman tulostaulun luominen QC-moduulin tietokantarakenteeseen. Tämä tekee tietokantaosan ohjelmakoodista monimutkaisemman, koska tuloksia voidaan joutua hakemaan kahdesta eri taulusta. Toinen vaihtoehto olisi ollut kopioida kaikki laadunvarmistustulokset tähän tauluun, mutta silloin laadunvarmistustulokset olisivat olleet kahdessa eri paikassa samassa tietokannassa, joka ei ole hyvää tietokannan käyttöä.

6.2.5 Diagrammien ominaisuuksien tallentaminen

Toteutusvaiheen ollessa jo pitkällä huomattiin, että eri laadunvalvontakorttien ominaisuuksien tallentamista ei oltu ajateltu riittävästi. Alkuperäinen idea oli tallentaa laadunvalvontakorttien ominaisuudet (kuten nimi, tyyppi, valitut valvontarajat, jne.) saman taulun eri sarakkeisiin. Näin jokaiselle ominaisuudelle olisi ollut oma sarake, ja tietokantaa olisi hyödynnetty tehokkaasti. Tämä malli havaittiin kuitenkin toimimattomaksi, sillä eri ominaisuuksia tuli korteille jatkuvasti lisää ja oli odotettavissa, että tulevaisuudessa niitä tulisi vielä enemmän. Jo pelkästään QC-moduulin versioon 1.0.0 tuli yli sata kappaletta laadunvalvontakorttien ominaisuuksia.

Tältä pohjalta suunniteltiin uusi tietokantataulu, joka sisälsi ominaisuuksia laadunvalvontakorttikohtaisesti. Taulun jokainen rivi määrittää yhden ominaisuuden

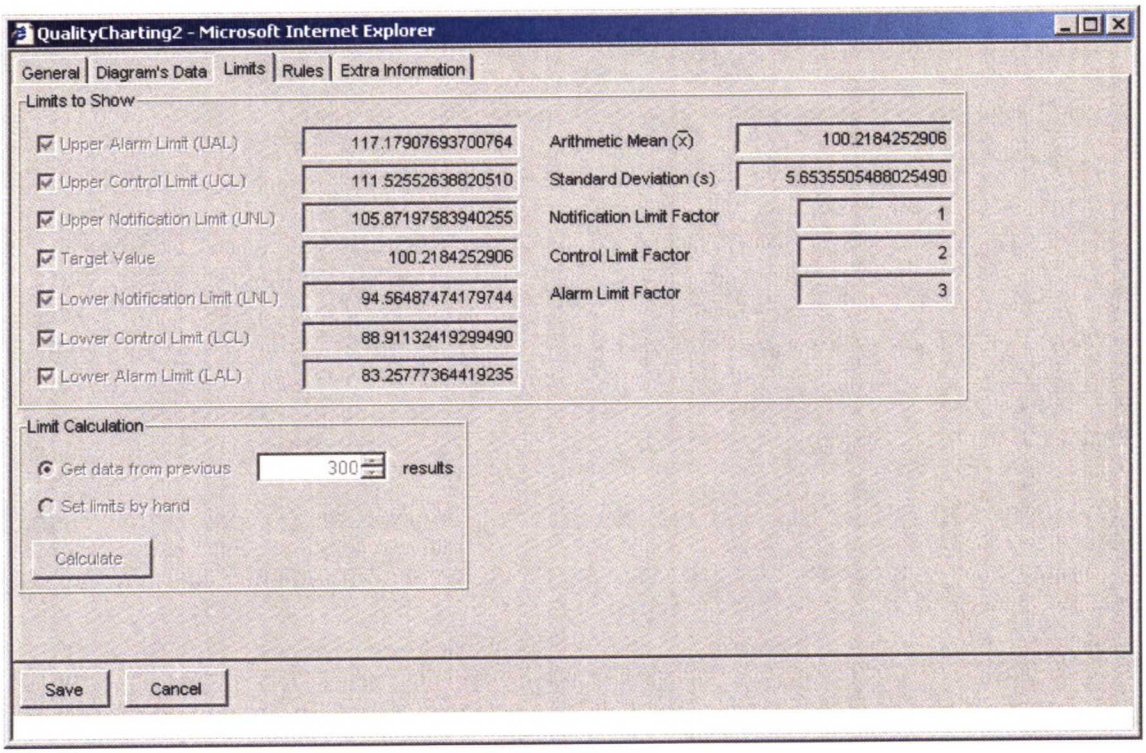
yhdelle kortille. Jokaisella rivillä on tieto, mihin laadunvalvontakorttiin se kuuluu. Varsinainen tieto voi olla tekstiä, numero tai päivämäärä. Näille kaikille tietotyypeille on oma sarakkeensa. Yksi sarake määrää mihin tietotyyppiin kyseinen ominaisuus kuuluu. Ominaisuudet on ryhmitelty avainkentän mukaan, esimerkiksi:

Avain	**Tyyppi**	**Tieto**
general.title	test_01	Teksti
general.creationdate	12.04.2004 13:44:20	Päivämäärä
basicstatistics.ual	100,576	Luku
basicstatistics.meanvalue	99,567	Luku

Taulukko 1 Esimerkkitaulukko laadunvalvontakortin ominaisuuksista

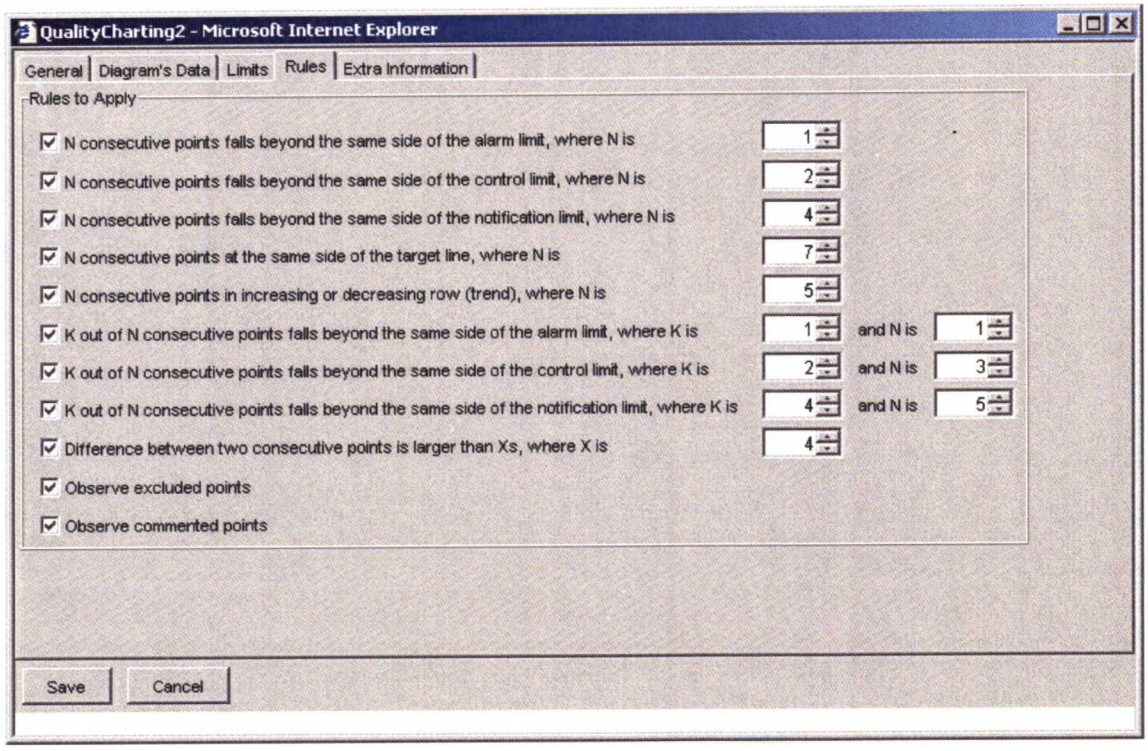
Näistä kaksi ensimmäistä määrittää laadunvalvontakortin käyttäjälle näkyvän nimen ja luontipäivän. Kaksi seuraavaa liittyy tilastollisiin tietoihin, ylempään valvontarajaan ja keskiarvoon. Ominaisuuksien tallennusmuodon muuttaminen merkitsi käytännössä koko tietokantaosan uudelleen ohjelmointia.

Oheiset kuvat näyttävät osan käyttäjälle näkyvistä ominaisuuksista (Kuva 10). Kyseisessä ikkunassa lasketaan laadunvalvontakortin rajat ja muut tilastolliset tunnusluvut. Esimerkkikuvassa kortti on Active-tilassa, joten valvontarajoja ei voi enää muuttaa.



Kuva 10 Laadunvalvontakorttien rajojen laskentanäyttö

Seuraavassa kuvassa (Kuva 11) on esimerkki valvontakortin sääntöjoukon määrittämisestä. Kyseisessä valvontakortissa käyttäjä on aktivoinut kaikki säännöt.



Kuva 11 Laadunvalvontakortin sääntöjoukon määrittämisnäyttö

6.3 Komponentin loppuarviointia

Komponenttia esiteltiin useita kertoja eri tahoilla sekä toteutusvaiheessa että version 1.0.0 julkaisemisen jälkeen. Aluksi QC-moduulia esiteltiin yksittäisille Software Pointin työntekijoille. Projektin loppuvaiheessa pidettiin suurempi esittelytilaisuus lähes koko Software Pointin Suomen toimiston henkilökunnalle. Software Pointin henkilöstö ja seurantaryhmä olivat tyytyväisiä QC-moduuliin. Ensimmäisen version julkaisemisen jälkeen myös muutamat asiakkaan edustajat saivat henkilökohtaisen esittelyn QC-moduulista. Myös tällöin palaute oli positiivista, joskin silloin esiintyi enemmän toivomuksia asiakaskohtaisista muutoksista, joita ei oltu toteutettu.

QC-moduulia esiteltiin myös suurehkolle ryhmälle Software Pointin asiakkaita. Tätä varten pidettiin erillinen työpajatilaisuus, johon kutsuttiin asiakkaitten edustajia kahdeksasta yrityksestä. Mukana oli asiakkaita, joille Sapphire-järjestelmä oli jo toimitettu sekä asiakkaita, joille toimitus oli parhaillaan käynnissä.

Asiakkaiden palaute oli hyvin myönteistä. Tällöin tuli esille myös yksi selvä QC-moduulin etu verrattuna kolmannelta osapuolelta hankittuun laadunvalvontakorttien visualisointityökaluun (esimerkiksi Quality Analystiin). QC-moduuli voisi toimia reaaliaikaisesti Sapphire-järjestelmän rinnalla. Käytännössä tämä tarkoittaa sitä, että kun asiakkaan analyysilaitteelta (esimerkiksi alkuaineanalysaattorilta) tulee sarja tuloksia, joiden seassa on muutama laadunvalvontatulos, voi Sapphire-järjestelmä ilmoittaa QC-moduulin tarkistusten perusteella lähes välittömästi, onko syötetty sarja hyväksyttävä vai ei. Joillakin asiakkailla tämä nopeuttaa huomattavasti analyysiprosessia ja antaa selvää lisäarvoa LIMS-järjestelmän käyttöön.

Asiakkaat myös vertasivat QC-moduulia nykyisin käytössä oleviin laadunvalvontakorttien visualisointiohjelmistoihin. Verrattuna esimerkiksi Quality Analystin kaltaiseen tilasto-ohjelmistoon, useimmat asiakkaat pitivät erittäin positiivisenä seikkana QC-moduulin räätälöintiä heidän tarpeisiinsa. Quality Analyst on heidän mielestään hyvä ohjelma, jos tehdään paljon monimutkaista tilastollista tutkimusta, mutta harvassa kemiallisessa laboratoriossa sitä kuitenkaan tehdään. QC-moduuli taas tekee juuri sen mistä asiakkaiden kanssa on sovittu. Työpajatilaisuuden

jälkeen eräs asiakas sanoi: "Olemme käyttäneet kolmannen osapuolen tilastollista ohjelmistoa laadunvalvontakorttien visualisointiin, mutta mielestämme kyseinen ohjelmisto sopii paremmin tilastotieteen tutkijoille tai suurkäyttäjille, mutta äsken näkemämme QC-moduuli tekee juuri sen mitä pitää: näyttää laadunvalvontavalvontakortteja. QC-moduuli siis vaikutti erinomaiselta meidän tarpeisiin."

Toisaalta, tietyt asiakkaat eivät nähneet hyötyä ottaa QC-moduulia käyttöön, koska heillä oli vain muutamia laadunvalvontakortteja hallittavana. Joillakin asiakkailla oli lisäksi yksittäisiä toivomuksia erikoisista toiminnallisuuksista, joiden puute esti QC-moduulin käyttöönoton. Näiden toiminnallisuuksien toteuttamisen työmäärän suhde toiminnallisuutta haluavien asiakkaiden lukumäärään katsottiin olevan liian suuri.

Vaikka palaute olikin lähes pelkästään positiivista, tulevaisuudessa on kuitenkin tarkoitus tehdä tarkempi kartoitus asiakkaiden mielipiteistä QC-moduulin osalta. Samalla voitaisiin kerätä parannusehdotuksia ja vinkkejä tuleviin QC-moduulin versioihin.

QC-moduulia on asennettu asiakkaiden käyttöön Suomessa, Ruotsissa ja Tanskassa. Ruotsissa ja Tanskassa on törmätty erikoisiin päivämäärä- ja lukuformaattiongelmiin, vaikka QC-moduulista yritettiin tehdä maa-asetusriippumaton. Tätä kirjoitettaessa nämä ongelmat ovat vielä tutkimuksen alla. Alustavien tietojen mukaan ongelma johtuu tietokannan ja QC-moduulin välissä olevan Sapphire-järjestelmän komponentin yhteensopimattomuusongelmista QC-moduuliin nähden. Suurta haittaa näistä ongelmista ei ole vielä ollut, sillä kyse on ollut maakohtaisesta testauksesta ennen asiakastoimituksia.

QC-moduulin tulevaisuus näyttää mielenkiintoiselta. Useimmat asiakkaat tarvitsevat laadunvalvontakorttien visualisointityökalua. Tällöin yksi vaihtoehto on QC-moduuli. Jatkossa tullaan tarkkailemaan, millä perusteilla asiakkaat visualisointityökalunsa valitsevat. QC-moduulin osalta kannattaisi myös pyrkiä tuotteen jatkuvuuteen. Asiakkaat voisivat olla kiinnostuneita päivittämään QC-moduulin tuleviin, paranneltuihin versioihin, jos QC-moduuliin saadaan sen arvoa

kasvattavia toiminnallisuuksia. Näiden toiminnallisuuksien löytämiseksi tulee tehdä yhteistyötä asiakkaiden kanssa.

Yhteenveto

QC-projekti oli helppo aloittaa, sillä jo etukäteen tiedettiin, että useilla asiakkailla on tarve laadunvalvontakorttien visualisointityökalulle. Projektin määrittelyvaiheessa käytettiin runsaasti aikaa asiakkaiden tarpeiden kuuntelemiseen, jotta QC-moduulista saataisiin asiakkaiden haluama työkalu. QC-moduulin suunnitteluvaiheessa vaatimuksia kerättiin kuitenkin useista eri lähteistä. Erityistä huomiota kiinnitettiin laadunvalvontakorttien teoreettisen pohjan ymmärtämiseen. Tässä käytettiin apuna kirjallisuutta ja asiakaskeskusteluissa saatua ymmärrystä. Huomattiin, että kirjallisuudessa oli vain hajanaisia mainintoja laadunvalvontakorttien käytöstä kemiallisissa laboratorioissa. Laadunvalvontakorttien käyttö kemiallisissa laboratorioissa eroaa merkittävästi tuotantoprosesseissa käytettävistä laadunvalvontakorteista. Lisäksi laadunvalvontakorttien käytössä oli asiakaskohtaisia eroja. Toteutusvaiheessa luotiin ensin prototyyppi, joka jätettiin käyttöön ja kehitettiin eteenpäin. Toteutus- ja suunnitteluvaihe linkittyivät voimakkaasti toisiinsa. Syklissä yleensä suunniteltiin pieni toiminnallisuus, joka sitten toteutettiin. Lopulta prototyypistä kasvoi ensimmäinen julkaistu QC-mduulin 1.0.0 -versio.

Projektin lopussa QC-moduulia esiteltiin useille eri tahoille. Yhtenä projektin onnistumisen merkkinä voidaan pitää sitä, että projektin lopussa asiakkailta tuli lähes yksinomaan hyvää palautetta. Suunnitelmissa on tehdä asiakastyytyväisyyskysely QC-moduulia käyttäneiden asiakkaiden keskuudessa. Tämän kyselyn pohjalta voitaisiin tulevaisuudessa kohdentaa moduulin jatkokehitystä ja viimeistelyresursseja.

Kokonaisuutena ajatellen QC-projekti onnistui erinomaisesti. Projektin tuloksena oli ohjelmisto, joka tuotteistettiin, joka ja on käytössä useilla eri asiakkailla. Kaikenkaikkiaan QC-moduulin tulevaisuus näyttää valoisalta.

VIITTEET

[Asaka] Asaka, T., Ozeki, K., Handbook of quality control tools, the Japanese approach, Productivity Press, Oregon, USA, 1990, s. 205-239

[Asiakaskeskustelut] Keskustelut laadunvalvontakomponentin asiakkaiden kanssa vuoden 2003 aikana.

[Fenix] Wilnor AB, Functional specification for Fenix project FENIX 1.0. Quality Assurance, Version 1.2, Number: 5000-02/REQUIRE/LIMSASSU.DOC, 1997.

[Griffith] Griffith, G., Statistical Process Control Methods for Long and Short Runs, Second Edition, ASQC Quality Press, 1996, s. 1-3, 13-35

[Kolarik] Kolarik, J., Creating quality: concepts, systems, strategies, and tools, McGraw-Hill, Texas, 1995, s. 364-367

[Kuokkanen] Kuokkanen, K., LIMS-järjestelmän kehittäminen akkreditoidun kliinisen kemian laboratorion tarpeisiin, Diplomityö, Teknillinen Korkeakoulu, Espoo, 2003, s.2-4, 13-15, 42-43

[Montgomery] Montgomery, D. C., Introduction to Statistical Quality Control, John Wiley & Sons, Inc., 4th ed., USA, 2001, s.206-248

[Prichard] Prichard, F.E, Quality in the Analytical Chemistry Laboratory, John Wiley & Sons, West Sussex, 1997, s. 153-155

[QCpp] Software Point Oy, Quality Control Module – Project Plan, 2003.

[QCsds] Software Point Oy, Quality Control Module - System Design Specification, 2003.

[QCurs] Software Point Oy, Quality Control Module – User

Requirement Specification, 2003.

[Shewhart] Shewhart, W. A., Economic Control of Quality of Manufactured Product, D. Van Nostrand Company, Inc., New York, 1931, s.249-261, 273-277, 309-320

[Tuurala] Tuurala, T., Laadun Historia, http://www.kotiposti.net/tuurala/Laadun%20historia.htm, viitattu 18.10.2003.

[Westgard] Westgard, O, Barry, P., Hunt, M., Groth, T., A Multi-Rule Shewhart Chart for Quality Control in Clinical Chemistry, Clinical Chemistry, Vol 27, No. 3, 1981

[Wheeler92] Wheeler, D. J., Chambers, D.S, Understanding Statistical Process Control, SPC Press, Inc., 2nd ed., Knoxville, 1992, s.1-54.

[Wheeler95] Wheeler, D. J., Advanced Topics in Statistical Process Control, SPC Press, Inc., Knoxville, 1995, s.31-74

[Wise] Wise, S.A., Innovative Control Charting, American Society for Quality, Milwaukee, 1998, s. 15-25, 55-82

[WSP] Whitelake Software Point Oy, Ten years of laboratory information management 1992-2002

Liite A: Grafiikkakomponentin etsinnän tulokset

Alla olevassa taulukossa on esitetty tiivistetysti grafiikkakomponentin etsinnän tulokset. Kirjoittanut Anna Laukkanen.

Product	Made by	Found	Licenses	Price	Other
KavaChart Applet Solutions	Visual Engineering	http://www.ve.com/applet.html	Developer kit or source code edition	$299 or $995	Supports scaling/zooming
EasyCharts	Objectplanet	http://objectplanet.com/EasyCharts/	Software license: 1/developer, covers applet usage and end user client application usage or Source license: java source code, 1/developer having access to the source code.	$299 or $799	Comes with a JpegEncoder that you can use for saving the charts as jpeg images
Line Graph Applet	Sirius Computer Consultants	http://www.jpowered.com/graph_chart/index.htm	Professional License: Multiple Location license (buy once implement anywhere !) or Source Code Licence	$199,95 or $499,95	Does not seem to produce the type of point/graph needed
EspressChart	Quadbase	http://www.quadbase.com/eces/index.html	Server/Deployment License: allows you to deploy quadbase solutions in applets, applications, servlets, JSPs, etc. - generally in a server environment. Deployment licenses are priced per server CPU.	Contact sales rep.	
JClass Chart	Sitraka	http://www.sitraka.com/software/jclass		$1299	
JFreeChart	The Object Refinery	http://www.jfree.org/jfreechart/index.html	Documentation: single electronic copy or Documentation: unlimited electronic copies at one site	US$39,95 or US$249,95	
JClass Chart	Quest Software	http://java.quest.com/jclass/chart.shtml		Contact	

Taulukko 2 Tiivistetty tuloslista grafiikkakomponentin etsinnästä

Liite B: Grafiikkakomponentin etsinnän tarkempi kuvaus

Seuraavassa on yhteenveto grafiikkakomponentin etsinnästä. Kirjoittanut Anna Laukkanen.

Java chart components

The reason for this study was to find suitable chart components for Java applets containing different types of data. Many different packages of chart components are available both on sale and as freeware. The problem is to find a set of components that is at the same time easy to use and highly customizable to meet our specific needs.

Besides the basic features present in all chart solutions, some of the most important features needed are zooming, giving different looks (colors, symbols) to points on the same line in a line graph, conversions into images and printing.

Chart examples

Visual Engineering, KavaChart Applet Solutions

Web site: http://www.ve.com/applet.html.

The package contains a large amount of Java classes, dealing with different types of charts, the axes and the data itself. The structure of the solution seems logical. Since the elements are separated into their own classes, the charts should be relatively easy to customize.

The possibility of zooming is given. Its functioning is not explained in the evaluation package, but an example is shown on the web site, where the user can draw a rectangle into which the zooming is done.

Testing the evaluation package was not very easy, since the Javadoc documentation only comes with the purchase.

Prize: developer kit $299, source code edition $995

ObjectPlanet, EasyCharts

Web site: http://objectplanet.com/EasyCharts/.

The different types of graphs are made as separate Java classes, but no classes exist that deal e.g. with the axis or the data. These features are coded into the graph classes themselves, and thus hard to access for further customization. The graphs are customizable using methods or applet properties, but it seems that adding new features would be difficult to accomplish.

From the web site: "The java chart library supports charts with multiple data series, overlay charts, drilldown charts, and interactive features such as zooming and scrolling of chart data."

Prize: Software license $299, Source license $799

Quest Software, JClass Chart

Web site: http://java.quest.com/jclass/chart.shtml.

Only jar files are given in the evaluation pack, so it is hard to say anything about the architecture of the system

According to the web site, the charts are easily customizable (appearance, axes, etc.)

From the web site: "The built-in support for a JavaBean customizer allows JClass Chart objects to be manipulated at design time or run-time and lets you modify and save charts using the chart applet HTML format." "With JClass Chart's map and pick functions you can enable applications to provide feedback when the user clicks the mouse anywhere on a graph. Transpose, drill down, edit and scale with any chart type to give your users the ultimate in interactive capabilities."

Prizes not given (need to contact sales).

The Object Refinery, JFreeChart

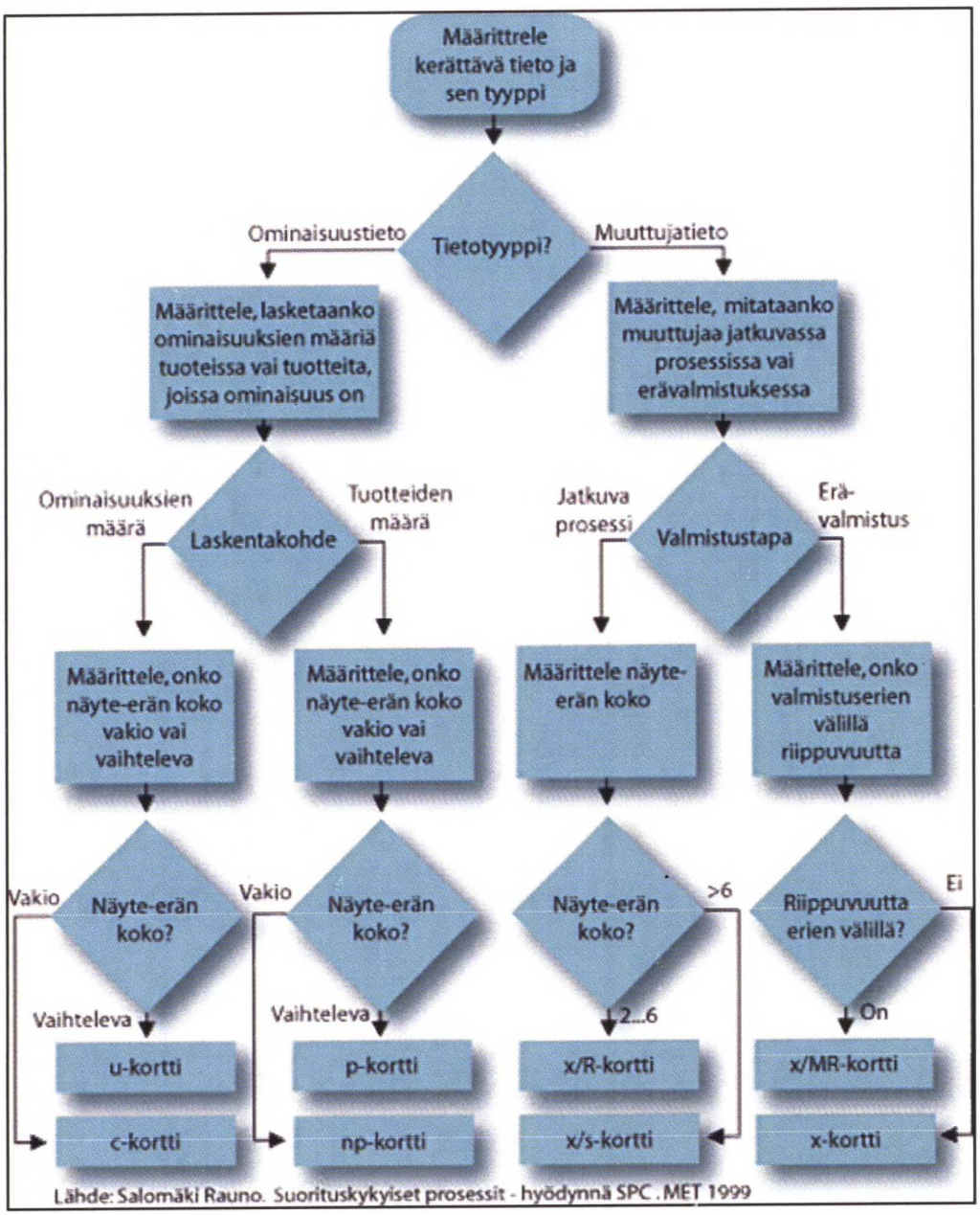
Web site: http://www.jfree.org/jfreechart/index.html

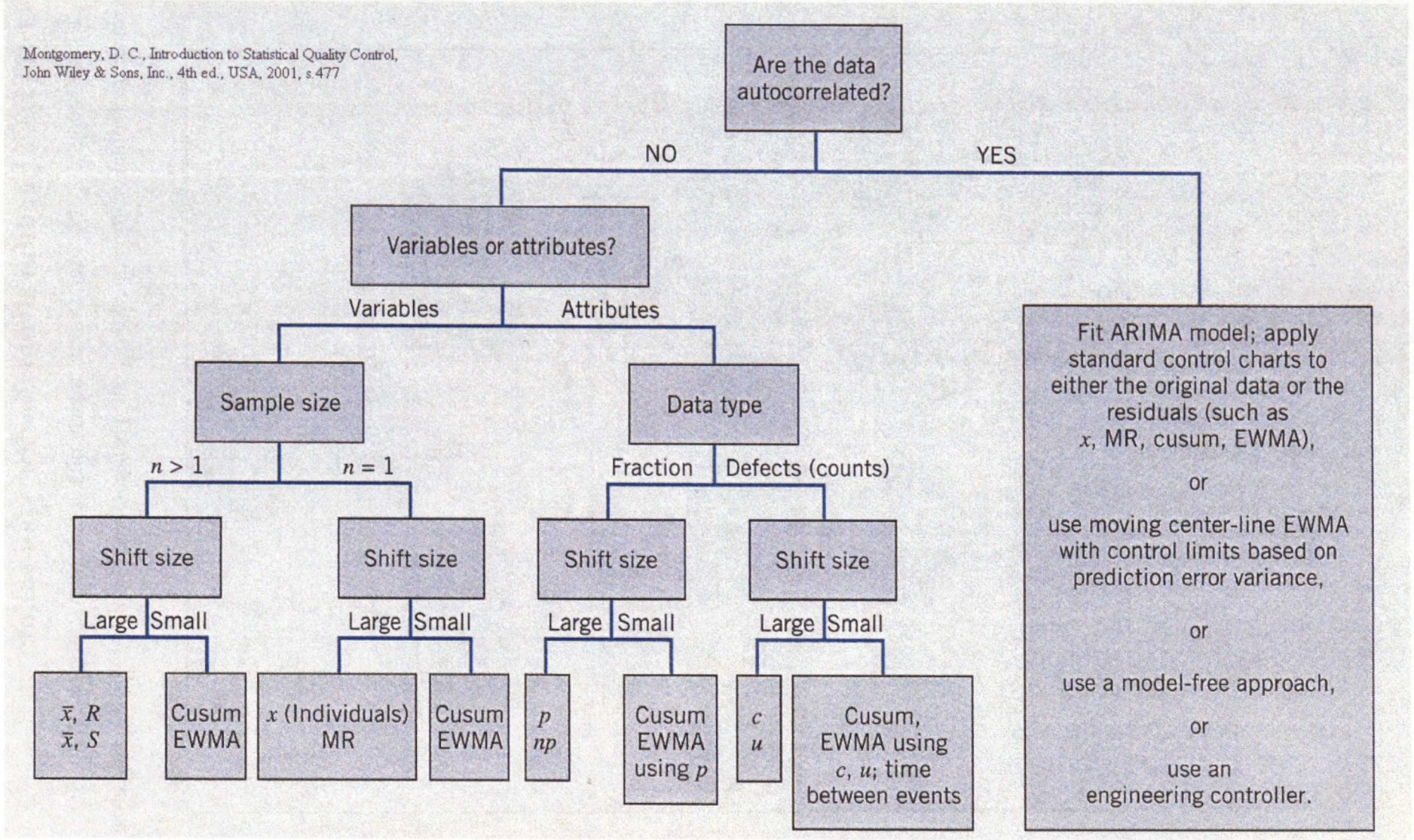
This freeware chart component package seems quite well done, and it offers most of the important functionalities needed, such as zooming, printing and saving in various formats.

The documentation for the package is on sale, with the prize of $39.95 for a single electronic copy and $249.95 for unlimited electronic copies at one site.

LIITE C: Laadunvalvontakorttityypin valinta

Kirjallisuudesta löytyy useita valmiita kaavioita, jotka auttavat löytämään oikean laadunvalvontakorttityypin eri prosessityypeille. Ohessa on listattu niistä muutamia.




Montgomery, D. C., Introduction to Statistical Quality Control,
John Wiley & Sons, Inc., 4th ed., USA, 2001, s.477
Are the data autocorrelated?
NO
YES
Variables or attributes?
Variables
Attributes
Sample size
Data type
$n > 1$
$n = 1$
Fraction
Defects (counts)
Shift size
Shift size
Shift size
Shift size
Large
Small
Large
Small
Large
Small
Large
Small
$\bar{x}$, R
$\bar{x}$, S
Cusum
EWMA
x (Individuals)
MR
Cusum
EWMA
p
np
Cusum
EWMA
using p
c
u
Cusum,
EWMA using
c, u; time
between events
Fit ARIMA model; apply standard control charts to either the original data or the residuals (such as x, MR, cusum, EWMA),
or
use moving center-line EWMA with control limits based on prediction error variance,
or
use a model-free approach,
or
use an engineering controller.

LIITE D: VAATIMUSMÄÄRITTELY

Tässä liitteessä on QC-projektin tiivistetty vaatimusmäärittely. Täydellinen vaatimusmäärittely löytyy QC-projektissa tehdystä *9017 User Requirements Specification* –dokumentista.[QCurs]

General requirements

The following table describes the general requirements.

Item	Description	Pri
GENERAL		
GEN-01	QC-module shall be Applet based.	1
GEN-02	QC-module Applet shall work in the Sapphire LVX and in a Web application.	1
GEN-03	QC-module shall work with the Oracle and SQL Server 2000 –databases.	1
GEN-04	QC-module shall be Swing based.	1
GEN-05	Translatable, foreign language support.	1
GEN-06	The data part and the viewing part shall be separated in the implementation.	1
GEN-07	Role level access by Sapphire, e.g. *qc_modul_edit*, *qc_modul_view* and *qc_modul_create*. With this role level access, it is possible to allow some people to create, destroy, view, etc. new diagrams.	1

Materials and QC Sample requirements

The following table describes the required functions for the Materials and QC Samples.

Item	Description	Pri
QUALITY SAMPLE		

Item	Description	Pri
QSA-01	It shall be possible to retrieve the quality control data from e.g. the MATERIAL table in the Sapphire database. However, this retrieving shall be configurable to retrieve different information from different locations (depending on customer's specification to store the QC-data in the database). For example there are many names (terms) for linking the specific QC-card to the database with unique identifiers e.g.: material, analysis, parameter, batch number, job number, instrument, test, attribute, product, etc. Because this depends on the customer needs, the QC-module can't be totally customer-independent.	1
QSA-02	It shall be possible to exclude one or multiple samples from the calculation. When excluding sample from the calculation one have to insert some comment on it.	1
QSA-03	Quality control result attributes in the database (in **bold** means obligatory): 1. **sequence number** 2. **date**, one millisecond precision 3. **calculated**, (yes or no), if included in the statistical calculations 4. **visible**, (yes or no), if visible in the diagram, if quality control result is not visible it can not be included in the calculatations either. 5. at least value 1 and value 2 (for R-diagrams), but there can be N quality sample results in parallel 6. comment (obligatory when excluding point from the calculations) However, these attributes have to be configurable (to add or remove attributes), because different customers have different systems and needs.	1
QSA-04	Module shall use Sapphire templates for diagram specific (properties) data. This helps e.g. when the user creates several new diagrams with almost same properties.	1
QSA-05	It shall be possible to link the diagram to the database by the pre-defined list of queries for linking the diagram data to the database data. Query attributes can be obtained (from the user) with the same kind of methods than in the Sapphire Web interface.	3

Requirements for limits and rules

The following table describes the required functions for the limits and rules.

Item	Description	Pri
DIAGRAMS		

Item	Description	Pri
ALA-01	There can be only one pair of 1s, 2s, 3s and one target line. 1s, 2s, 3s and target line shall be marked with a different color and style.	1
ALA-02	It shall be possible to create any number of user defined limits. The user can select the color and the style for the limits. However, the user defined limits can not contain any alarm functionality (e.g. showing PP).	1
ALA-03	It shall be possible to set 1s, 2s, 3s, standard deviation and target line – values manually.	1
ALA-04	It shall be possible to retrieve 1s, 2s and 3s from the Sapphire specifications.	1
ALA-05	It shall be possible to calculate standard deviation and mean values by: 1. calculated from N consecutive quality control results (can be selected by sequence number or by date interval) 2. calculated from N quality control results before the newest result (N latest results)	1
ALA-06	It shall be possible to set the rule active or inactive. This enables use of a different kind of run-rules-set, e.g. Shewart, Westgard, AT&T, Motorola, etc. As default all rules are active. See the Figure 1 for the procedure of the rule running order.	1
ALA-07	**N>3S**: 1 or more (default 1) consecutively points falls beyond the same side of 3s, then show PP.	1
ALA-08	**N>2S**: 1 or more (default 2) consecutively points falls beyond the same side of 2s, then show PP.	1
ALA-09	**N>1S**: 1 or more (default 4) consecutively points falls beyond the same side of 1s, then show PP.	1
ALA-10	**N>TAR**: N consecutively points in the same side of the target line, then show PP. (N > 0).	1
ALA-11	**N>ROW**: N consecutively points in an increasing or decreasing row (trend), then show PP. (N > 0).	1

Item	Description	Pri
ALA-12	**KN>1S**: K out of N (default 4 out of 5) consecutive points fall beyond the same side of 1s, then show PP. (K > 0) and (N > 0).	1
ALA-13	**KN>2S**: K out of N (default 2 out of 3) consecutive points fall beyond the same side of 2s, then show PP. (K > 0) and (N > 0).	1
ALA-14	**KN>3S**: K out of N (default 1 out of 1) consecutive points fall beyond the same side of 3s, then show PP. (K > 0) and (N > 0).	1
ALA-15	**D>XS**: The difference X between two consecutive points is larger than Xs (multiply X by standard deviation), then show PP (default difference X is 4.0). (If we set the resolution to 0.1, then 0.2 is the next adequate value, so the minimum value that user can set is 0.2, i.e, X => 0.2.)	1
ALA-16	If point is excluded from the calculations, then show PP (if visible state is on).	1

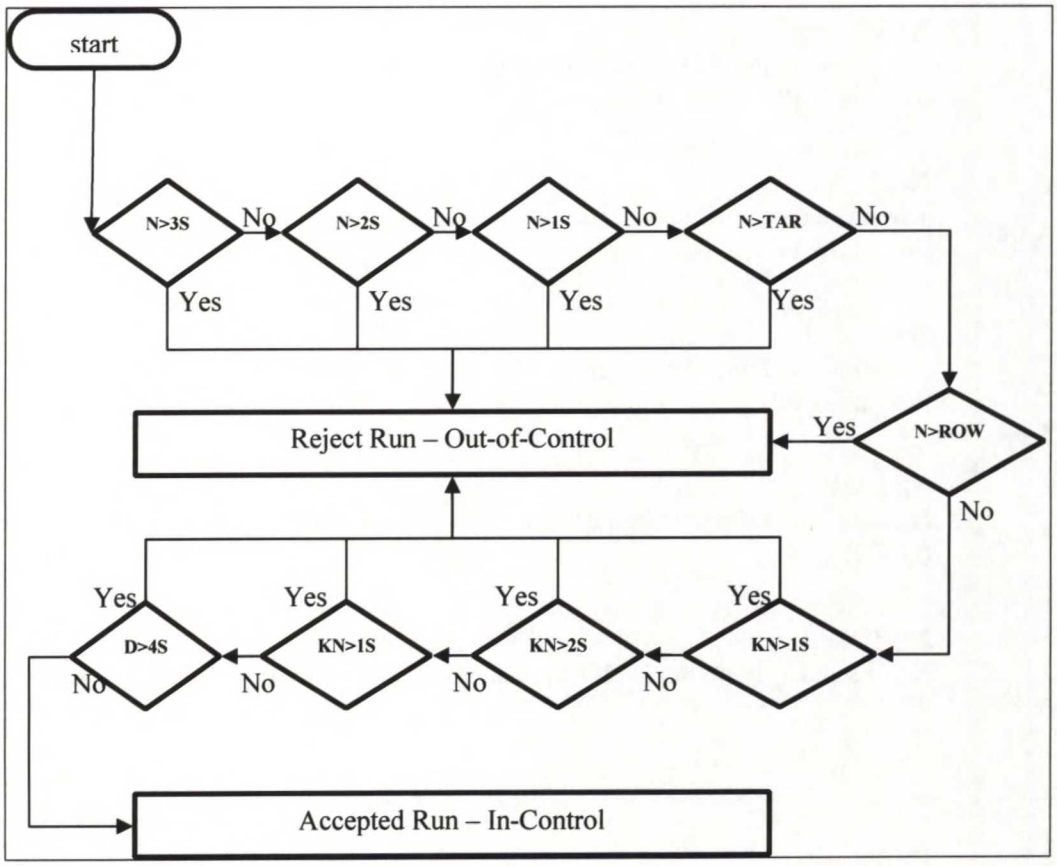


Figure 1 Procedure for running through the rules. Note: Westgard's model uses different run order, which is also supported.

User Interface Requirements

The following table describes the required functions for the user interface.

Item	Description	Pri
USER INTERFACE		
GUI-01	It shall be possible to see the x-, $\bar{x}$ -, r- and r-%-diagrams for the quality control result data. 1. **x-diagram**: target line is the mean value of user selected points, shows 1s, 2s, 3s limits and diagram's point value is a one single quality control sample result 2. $\bar{x}$ **-diagram**: same as the x-diagram, but diagram's point value is the mean value of N quality control sample result 3. **r-diagram**: target line is the zero value, shows 1s, 2s and 3s limits, diagram's point value is difference between quality control sample results, see the exact formula for difference from the *Mathematics* section, it should be possibly to view the diagram also from absolute i.e. intrinsic difference values 4. **r-%-diagram**: shows N relative-r-diagrams in the same diagram (one on the other, transparency style), diagram will calculate the relative-r-diagram from the given r-diagrams, see the exact formula for relative difference from the *Mathematics* section, no 1s, 2s or 3s limits	1

Item	Description	Pri
GUI-02	Diagram attributes that the user may change: for the x-, $\bar{x}$ -, r- and r-%-diagrams: 1. **name**, which is also an unique id in the database 2. **diagram title** 3. **unique identifiers**, identifies from where quality control result data is retrieved, these parameters can be e.g. table and column names in the Sapphire database (e.g. MATERIAL and ANALYSIS) 4. **state**, [COLLECTING, LOCKED, CLOSED], when in the collecting-state, the user can change 1s, 2s and 3s limits, but those properties can not be changed, when the locked-state is first time (and last time) set. This feature enables Diagram Immutable –property, which means that diagram is guaranteed to display the same information e.g. after five years. Closed state is reserved for future use. 5. **statemoddt**, date when diagram's state was last time changed. 6. **card type**, set when user creates the card, x, $\bar{x}$,r or r-% -type 7. **person in charge** 8. **y-axes label, color and style** 9. **x-axis by, date or sequence number** 10. **comment**, at least 250 chars 11. **creation date** 12. **PP's style and color** 13. **s1, s2, s3 and target line color and style** For r-%-diagrams, the y-axis label is % by default.	1
GUI-03	Data sequence. The user may define how many samples are shown in the diagrams by selecting time interval or sequence number interval (default shall be "N samples before the newest sample").	1
GUI-04	Data shall be displayed also in a table (e.g. below the diagram). This data shall include at least following information: 1. Exact quality control sample result(s) 2. Sequence number 3. Date and time 4. Difference of the quality control sample results 5. Status 6. Difference in s-units (standard deviation) 7. Comment However, the data table columns shall be configurable to show different information for different customers.	1

Item	Description	Pri
GUI-05	Statistical and other data. In the diagrams also some extra data must be seen. Statistical data, this data shall be displayed without the excluded points and with the excluded points: 1. **m**, arithmetic mean (or nominal, zero, target value) 2. **n**, number of results 3. **s**, standard deviation 4. $\mathbf{s_r}$, relative standard deviation 5. $\mathbf{s_m}$, standard deviation of the mean 6. $\mathbf{s_r(\%)}$, percentage relative standard deviation 7. $\mathbf{s^2}$, variance See the *Mathematics* section for mathematical explanations. Other data: 1. User id (the card creator and the current user) 2. 1s, 2s and 3s limits 3. User defined limits 4. PPs explanations (warning, excluded, …) 5. Number of excluded samples 6. Text: "x-diagram" where x can be x, $\bar{x}$, r or r-%. 7. Name of the diagram 8. Diagram's person in charge 9. Diagram's creation date 10. The diagram's comment 11. Sequence numbers or dates of points (quality control results) that the calculations have been made	1
GUI-06	It shall be possible to view result data from a single point in the graph (e.g. moving the mouse pointer over one point). This data shall be: 1. Sequence number 2. Date 3. Exact quality control sample result(s) 4. Difference between quality control sample results 5. Person in charge 6. Status 7. Comment Note, some of these can be empty (not-shown). However, this data shall be configurable to show different information for different customers.	1

Item	Description	Pri
GUI-07	User interface shall be 'intelligent' so that it will minimize amount of mouse clicking in the basic operations. This can be achieved e.g. by saving the user interface positions (diagram's sizes and positions, selected options, typed texts, etc.), so that there is no need for 'meaningless' clicking when user starts the module next time.	1
GUI-08	It shall be possible to see the calculated parameters: 1. Calculated mean value line in the diagram 2. Calculated 1s, 2s and 3s value lines in the diagram These values shall be calculated from all result points included in the visible diagram.	1
GUI-09	It shall be possible to open a simple x-, $\bar{x}$ - or r-diagram from e.g. some other applet or from the test-result-input -page in Sapphire. This opening shall be possibly: 1. by giving all the parameters in the call: opening new diagram 2. by giving the id of the diagram: opening existing diagram	1
GUI-10	It shall be possible to select which ones of the 1s, 2s and 3s limits are shown in the diagram.	1
GUI-11	It shall be acceptable to have blank results in the calculations. Blank results are ones that have a 'place' for the result in the database, but does not contains the actually result value.	1
GUI-12	It shall be possible to scale the x- and y-axis in the diagram (zooming). After the scaling all data in the window must be refreshed, so that there will be no situation, when e.g. the statistics calculated from the diagram are calculated from the old (before the zooming) diagram.	1
GUI-13	It shall be possible to edit card's properties easily e.g. just by pressing Edit button in the diagram window.	1
GUI-14	It shall be possible to see what is the exact rule that certain PP breaks.	1
GUI-15	It shall be possible to print the diagrams, related information (statistical data) and the table data, separately or combined Prints should comfort ISO-17025 standard.	1

Item	Description	Pri
GUI-16	It shall be possible to edit (only, all other edit functionality shall be done by diagram's property page or by Sapphire's other operations) point in the diagram window: 1. Removing point from the calculation 2. And when the point has been removed, then comment have to be set	1
GUI-17	It shall be possible to refresh the data in all diagram displays with a one button click.	1
GUI-18	It shall be possible to set automatic refresh-operation, i.e., the diagram display is refreshed after N minutes.	3
GUI-19	It shall be possible to copy (copy-and-past –functionality) any user selected rectangle area from the diagram windows.	3
GUI-20	It shall be possible to restore the diagrams in some specific graphics format (jpg, gif) into the database.	3
GUI-21	It shall be possible to remove a point from the diagram (point not visible). This is useful e.g. when the user (or some laboratory machine) accidentally feeds a incorrect result (e.g. in totally different scale) in the database. This kind of result can spoil the diagram. However, this 'remove from the digram' won't remove the result from the database.	1
GUI-22	Sound (e.g. beeb) when new point is added to the diagram and it breaks some rule. Usefull when user attention is needed, e.g. some fast paced process.	3

Other Requirements

The following table shows the other main requirements of the project:

Requirement	Comments
Quality Control	The project shall include all standard quality control issues, such as versioning, change control and document control.

Requirement	Comments
Testing	Will be done according to test plans. There are some important testing procedures included in user requirements too. Heuristic user interface testing with at least one outside-user but if possible, some real laboratory user testing shall be conducted. Performance testing (e.g. with a huge databases).
Documentation	The configuration, its usage, and implemented tool shall be documented in English. Also a general introduction to the Quality Control Diagrams shall be done.
Architecture	Adding new functional modules for different calculations (e.g. multivariate) shall be easy.

Mathematics

This chapter includes mathematical schemas for the QC-module.

Id	Formula	Explanation
FOR-01	$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$	**Mean, average value** The sum of a series of observations divided by the number of observations. The average of all observations. If sample is random then $\bar{x}$ is the best estimate of the population mean.

FOR-02	$$s = \sqrt[2]{\frac{n\sum_{i=1}^{n} x_i^{\,2} - \left(\sum_{i=1}^{n} x_i\right)^2}{n(n-1)}}$$ $$s = \sqrt[2]{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$ $$s = \sqrt[2]{\frac{\sum_{i=1}^{n} x_i^{\,2} - n\bar{x}^2}{n-1}}$$	**Standard deviation** The positive square root of the variance. The sample estimate of the population standard deviation. Standard deviation is a measure how widely values are dispersed from the average value (the mean). The minimum number of observations necessary to obtain a useful estimate of the standard deviation is about six. Chebyshev's rule for any distribution (skewed, symmetric, bi-modal, ...): • at least 0 % of all observations within 1s • at least 75 % of all observations within 2s • at least 88,89 % of all observations within 3s There are many different ways to calculate standard deviation, which may result to a different output on a different computer (naturally, exact mathematical outputs equals). This may happen because of the computer's limited calculation capacity of decimal parts of numbers. This Quality Control module uses unlimited decimal part precision (in the limits of computer memory) for the calculations, so selecting the right formula is not essential. See also the variance formula.
FOR-03	$$s_r = RSD = \frac{s}{\bar{x}}$$	**Relative standard deviation** The relative standard deviation of a variable. This is a measure of the spread of data in comparision to the meanof the data. Sometimes this is confused with the percentage relative standard deviation.

FOR-04	$s_r(\%) = C_v = RSD \times 100\%$ $s_r(\%) = \frac{s}{\bar{x}} \times 100\%$	**Percentage relative standard deviation, coefficient of the mean** (not recommended) The percentage relative standard deviation of a variable.
FOR-05	$s_m = s_{\bar{x}} = \frac{s}{\sqrt[2]{n}}$	**Standard deviation of the mean, standard error of the mean** The standard deviation of the mean or standard error of the mean. The sample (experimental) standard deviation of the mean, also known as the standard error. It is an estimation of the variation that would arise if repeated samples were taken from the population.
FOR-06	Sample variance $s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$ Population variance $s^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}$	**Variance** The unbiased sample estimate of the population variance. The variance measures the extent to which the result differ from each other , the larger the variance the greater the spread of the data. The divider $n-1$ should be changed to N if you are ever in the situation of measuring the variance of a distribution whose mean $\bar{x}$ is known a priori rather than being estimated from the data. Then variance isn't any more an estimate and it is marked as $\sigma^2 = \frac{\sum(x-\mu)^2}{N}$ where μ is the population mean and N is the population size, then standard deviation is $\sigma = \sqrt[2]{\frac{\sum(x-\mu)^2}{N}}$. See also the standard deviation formula.
FOR-07	$d = \max(A) - \min(A)$	**Difference for r-diagrams** From a group A of parallel quality control results, the difference d can be calculated as formulated.

FOR-08	$d\% = \frac{\max(A) - \min(A)}{avg(A)} \times 100\%$	**Difference for r-%-diagrams** From a group A of parallel quality control results, the relative difference d% can be calculated as formulated.